WebSphere® IBM WebSphere Telecom Web Services Server

Version 7.1

# IBM WebSphere Telecom Web Services Server

**Third Edition (April 2010)**

This edition applies to IBM WebSphere Telecom Web Services Server, Version 7.0, and to all subsequent releases and modifications until otherwise indicated in new editions.

A form for readers' comments appears at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
Department 6R4A
P.O. Box 12195
Research Triangle Park, North Carolina
27709-2195

# Contents

Chapter 1. Introduction to WebSphere Telecom Web Services Server . . . . . 1
New features . . . . . . . . . . . . . . 1
Telecom Web Services Access Gateway . . . . . 5
Service Policy Manager . . . . . . . . . . . 7
Service Platform components . . . . . . . . . 9
  Admission Control component Web service . . . 9
  Traffic Shaping component Web service . . . . 10
  Network Resources component Web service . . 10
  Notification Management component Web service 10
  PX Notification component Web service . . . . 11
  Address Masking component Web service . . . 12
  Fault and Alarm component Web service . . . 14
  Usage Record component Web service . . . . 14
  Integrating with a Privacy Web Service . . . . 14
Web service implementations . . . . . . . . 14
  Web service implementations based on SIP/IMS 15
  Web service implementations based on IBM WebSphere software . . . . . . . . . . 16
  Web service implementations based on Direct Connect protocols . . . . . . . . . . . 16
  Web service implementations based on Parlay . . 17

Chapter 2. Planning to install WebSphere Telecom Web Services Server . . . . . . . . . . . . . . . . 19
Hardware and software requirements. . . . . . 19
  Hardware requirements . . . . . . . . . 19
  Software requirements. . . . . . . . . . 22
Evaluating your hardware environment . . . . . 24
Standard configurations for Telecom Web Services Server . . . . . . . . . . . . . . . . 25
Planning to install the Access Gateway . . . . . 27
Planning to install the Service Policy Manager. . . 28
Planning to install the Service Platform components 28
Planning to install Web service implementations . . 30
  Integrating the Web service implementations with other components . . . . . . . . . 30
  Planning checklist for Web service implementations. . . . . . . . . . . . . 33
  Planning for Web service implementations based on SIP/IMS . . . . . . . . . . . . . 42
  Planning for Web service implementations based on IBM WebSphere software. . . . . . . . 56
  Planning for Web service implementations based on Direct Connect protocols . . . . . . . . 61
  Planning for Parlay-based Web service implementations. . . . . . . . . . . . . 70
Planning for the database. . . . . . . . . . 87
  Database tables for Telecom Web Services Server subcomponents . . . . . . . . . . . . 88
  Database tables and stored procedures for Web service implementations . . . . . . . . . 92
Planning for security . . . . . . . . . . . 94
  General considerations for setting up security . . 95
  User groups and security roles . . . . . . . 97
  Planning authentication security using the Trust Association Interceptor . . . . . . . . . 98
Planning for notification. . . . . . . . . . 102
Planning to migrate from the previous version of Telecom Web Services Server . . . . . . . . 104
Worksheets . . . . . . . . . . . . . . 107
  Component planning worksheet . . . . . . 108
  Distributed database planning worksheet . . . 109

Chapter 3. Installing WebSphere Telecom Web Services Server . . . . 113
Telecom network integration scenarios . . . . . 113
  Installing Telecom Web Services Server on a single cluster . . . . . . . . . . . . 114
  Installing Telecom Web Services Server on multiple clusters . . . . . . . . . . . 115
Migrating from the previous version of Telecom Web Services Server . . . . . . . . . . . 117
  Migrating a non-production system to Telecom Web Services Server version 7.0 . . . . . . 117
  Migrating a production system to Telecom Web Services Server version 7.0 . . . . . . . . 119
  Migrating data in a distributed database configuration . . . . . . . . . . . . 121
  Changes to database tables for Telecom Web Services Server version 7.0 . . . . . . . . 122
  Completing the migration . . . . . . . 124
Preparing the environment . . . . . . . . . 125
  Preparing the operating system . . . . . . 125
  Preparing for the Telecom Web Services Server database . . . . . . . . . . . . . . 127
  Creating the clusters . . . . . . . . . . 129
Preparing to use the Trust Association Interceptor 140
  Preparing the installation files for the Trust Association Interceptor . . . . . . . . . 140
  Installing the Trust Association Interceptor security component . . . . . . . . . . 141
Installing Telecom Web Services Server base components . . . . . . . . . . . . . . 144
  Installing Telecom Web Services Server base components using the interactive installer . . . 144
  Installing Telecom Web Services Server base components silently . . . . . . . . . . 147
  Verifying the results after installing Telecom Web Services Server base components . . . . 152
Installing Telecom Web Services Server services 153
  Installing Telecom Web Services Server services using the interactive installer . . . . . . . 153
  Installing Telecom Web Services Server Web services silently. . . . . . . . . . . . 154
  Verifying the results after installing Telecom Web Services Server services . . . . . . . 158
Configuring your installed system . . . . . . 158

Creating and configuring the DB2 database server instance . . . . . . . . . . . . . 158
Creating and configuring the Oracle database server instance . . . . . . . . . . . . . 160
Running the First Steps configuration script . . 162
Configuring connections for the Web service implementations . . . . . . . . . . . 175
Verifying the completed configuration . . . . 186
WebSphere Application Server configuration for the Telecom Web Services Server components . 187
Adding a node or server Instance . . . . . 189
Installing updates for Telecom Web Services Server 190
Removing Telecom Web Services Server . . . . 191
Removing applications . . . . . . . . . 191
Removing databases and files . . . . . . . 194
Uninstalling the Trust Association Interceptor Security Component . . . . . . . . . . 195
Uninstalling the Telecom Web Services Server base components using the interactive installer . 196
Uninstalling Telecom Web Services Server base components silently . . . . . . . . . . 198
Uninstalling the Telecom Web Services Server services using the interactive installer . . . . 199
Uninstalling theWeb service implementations silently . . . . . . . . . . . . . . . 200

## Chapter 4. Configuring WebSphere Telecom Web Services Server . . . . 203

Configuring the Access Gateway . . . . . . . 203
Connecting the Access Gateway cluster to the Web service implementations cluster . . . . 203
Configuring additional settings for the Access Gateway . . . . . . . . . . . . . . . 204
Configuring the edge server . . . . . . . 206
Configuring the Service Platform and the Web service implementations. . . . . . . . . . 206
Configuring endpoints for Service Platform components and Web service implementations . 207
Configuring the Parlay Connector . . . . . 208
Configuring naming schemes . . . . . . . 243
Configuring privacy integration . . . . . . 245
Configuring communication with the SMSC . . 246
Enabling the Common Event Infrastructure Service . . . . . . . . . . . . . . . 248
Configuring a single network resource for service implementations. . . . . . . . . 254
Configuring integration with other parts of the network . . . . . . . . . . . . . . . 255
Provisioning the Inter Operator Identifier (IOI) 255
Finding the IBM XDMS XCAP root . . . . . 256

## Chapter 5. Administering WebSphere Telecom Web Services Server . . . . 257

Stopping and starting the server . . . . . . . 257
Stopping a cluster . . . . . . . . . . 257
Stopping a server (console). . . . . . . . 258
Stopping a server (command line) . . . . . 258
Stopping the node agent (console) . . . . . 259
Stopping the node agent (command line) . . . 259
Stopping the deployment manager (console) . . 260
Stopping the deployment manager (command line) . . . . . . . . . . . . . . . . 260
Starting the deployment manager . . . . . 260
Starting the node agents. . . . . . . . . 261
Starting a cluster . . . . . . . . . . . 261
Starting a server (console) . . . . . . . . 262
Starting a server (command line) . . . . . . 262
Restarting applications . . . . . . . . . . 263
Modifying logging . . . . . . . . . . . 263
Monitoring system performance using WebSphere PMI . . . . . . . . . . . . . . . . . 265
Enabling performance monitoring . . . . . 265
Disabling performance monitoring . . . . . 266
Performance metrics . . . . . . . . . . 266
Administering the Service Policy Manager. . . . 268
Launching the Service Policy Manager console 269
Using the features of the Service Policy Manager 270
Administering Service Platform components . . . 277
Administering the Admission Control component Web service . . . . . . . . . 278
Administering the Traffic Shaping component Web service . . . . . . . . . . . . . 279
Administering the Network Resources component Web service . . . . . . . . . 282
Using the Notification Management user interface . . . . . . . . . . . . . . . 283
Administering the PX Notification component Web service . . . . . . . . . . . . . 287
Administering the Fault and Alarm component Web service . . . . . . . . . . . . . 288
Disabling Service Platform components for Web services . . . . . . . . . . . . . . . 289
Administering Web service implementations . . . 289
Administering Web service implementations based on SIP/IMS. . . . . . . . . . . 290
Administering Web service implementations based on IBM WebSphere software . . . . . 294
Administering Web service implementations based on Direct Connect protocols . . . . . 296
Administering Parlay-based Web service implementations . . . . . . . . . . . 310
Maintaining the usage records database table. . . 317
Using IBM Tivoli License Manager . . . . . . 323

## Chapter 6. Integrating and Developing with WebSphere Telecom Web Services Server . . . . . . . . . . 325

Integrating with the Notification Management component Web service . . . . . . . . . 325
Notification Management implementation . . . 325
Removing notifications . . . . . . . . . 326
Interacting with SIP network elements . . . . . 327
Originating (initiating) the SIP request . . . . 327
Terminating (answering) the SIP request . . . 327
Custom JCA adapter . . . . . . . . . . . 328
Developing a custom JCA adapter . . . . . 329
JCA adapter functions . . . . . . . . . 331
JCA adapter records . . . . . . . . . . 332
Sample code to allow the Web service implementation to interact with the JCA adapter 335

Open network enablers API . . . . . . . . . 336
  Message flows . . . . . . . . . . . . . 336
  Interfaces . . . . . . . . . . . . . . 337
  Exceptions . . . . . . . . . . . . . . 341
Integrating with Web service implementations . . 342
  Integrating with Web service implementations based on SIP/IMS . . . . . . . . . . . 342
  Integrating with Web service implementations based on IBM WebSphere software . . . . . 351
  Integrating with Web service implementations based on Direct Connect protocols . . . . . 356
  Integrating with Parlay-based Web service implementations . . . . . . . . . . . 364
Integrating your applications with the Access Gateway . . . . . . . . . . . . . . . . 372
Access Gateway message processing flows . . . 372
  Using the Telecom Web Services Server plug-in for WebSphere Integration Developer . . . . 372
  Developing and customizing a custom Access Gateway flow . . . . . . . . . . . . 378
  Reference for Access Gateway message processing flows . . . . . . . . . . . 388
Programming an SPM MBean script . . . . . . 396
Migrating your applications from a previous release of Telecom Web Services Server . . . . . 398

## Chapter 7. Troubleshooting WebSphere Telecom Web Services Server . . . . . . . . . . . . . . . 401

Using ISA 4.0 add-ons to communicate with IBM Support . . . . . . . . . . . . . . . . 401
Troubleshooting Service Platform components . . 401
Character encoding for the TWSS Administration Console . . . . . . . . . . . . . . . . 403
Configuring a secondary Parlay gateway for failover . . . . . . . . . . . . . . . . 403
Troubleshooting WID. . . . . . . . . . . . 403
Monitoring log messages . . . . . . . . . . 404
  Viewing and modifying logs . . . . . . . 405
  Enabling trace . . . . . . . . . . . . 406
  Selecting trace loggers . . . . . . . . 408
Messages . . . . . . . . . . . . . . . . 409
  Message key. . . . . . . . . . . . . . 409
  Messages_SOAC_ESB_en_US . . . . . . . 410
  Messages_SOAC_en_US . . . . . . . . . 421
  Messages_en_US . . . . . . . . . . . 428
  Messages_SOAS_en_US . . . . . . . . . 477
  Messages_SOAX_en_US . . . . . . . . . 487
  Messages_en_US . . . . . . . . . . . 518
  Messages_en_US . . . . . . . . . . . 523

## Chapter 8. Reference information for WebSphere Telecom Web Services Server . . . . . . . . . . . . . . . 567

Telecom Web Services Server Standards and specifications . . . . . . . . . . . . . . 567
Default Web service URIs . . . . . . . . . 570
Reference: mediation primitives . . . . . . . 575
  Message Element Remover . . . . . . . . 575
  Transaction Identifier . . . . . . . . . 577
  Policy Retrieval. . . . . . . . . . . . 579
  Address Masking . . . . . . . . . . . 581
  Transaction Recorder . . . . . . . . . 584
  Network Statistics . . . . . . . . . . 587
  Message Interceptor . . . . . . . . . . 590
  Service Authorization. . . . . . . . . . 592
  Group Resolution . . . . . . . . . . . 593
  SLA Enforcement mediation primitives . . . . 595
  Service Invocation . . . . . . . . . . . 601
  JMX Notification . . . . . . . . . . . 603
  CEI Event Emitter . . . . . . . . . . . 605
Reference: default policies for the Access Gateway and Web service implementations . . . . . . 605
  Access Gateway . . . . . . . . . . . . 606
  Default policies for Web service implementations based on SIP/IMS . . . . . 606
  Default policies for Web service implementations based on IBM WebSphere software . . . . . . . . . . . . . . . 614
  Default policies for Web service implementations based on Direct Connect protocols . . . . . . . . . . . . . . . 615
  Default policies for Parlay-based Web service implementations . . . . . . . . . . . 625
  Changed policy names for migration . . . . 631
Reference: usage records for the Web service implementations . . . . . . . . . . . . 634
  Usage records for Web service implementations based on SIP/IMS . . . . . . . . . . . 635
  Usage records for Web service implementations based on IBM WebSphere software . . . . . 646
  Usage records for Web service implementations based on Direct Connect protocols . . . . . 649
  Usage records for Parlay-based Web service implementations . . . . . . . . . . . 666
  Attributes for the Notification Management component Web service . . . . . . . . . 684

## Chapter 9. Reference information . . . 687

Changes to this edition . . . . . . . . . . 687
Documentation conventions . . . . . . . . . 687
Directory conventions . . . . . . . . . . 687
Glossary . . . . . . . . . . . . . . . . 688

## Notices . . . . . . . . . . . . . . 699

Trademarks . . . . . . . . . . . . . . . 700

# Chapter 1. Introduction to WebSphere Telecom Web Services Server

IBM® WebSphere® Telecom Web Services Server allows you to expose high-level Web service interfaces to network services for third parties.

Third parties are typically telecommunications service providers or organizational divisions wanting to develop new services that integrate with their network infrastructures.

Web service interfaces provide access to service capabilities in a technology-independent way, using programming languages. Each Web service interface can have multiple backend implementations for connecting with a service provider's environment. For example, a Web service interface may connect to the network through the Session Initiation Protocol (SIP), the Parlay gateway using the Parlay Connector, the native protocols, or by using the custom integrated services.

## New features

IBM WebSphere Telecom Web Services Server (TWSS) version 7.0 includes functionality and support for new Web service implementations.

### Automated installation and configuration

Telecom Web Services Server version 7.0 includes automated installation routines for installing the base components and the Web service implementations. The routines can run interactively as wizards, or in silent mode using response files.

The version 7.0 product also provides a First Steps configuration script to ease the task of configuring the system. The system can be set to initialize, migrate policies and configurations from a previous release level, and update the system after fixes have been applied.

The First Steps script provides the following features:

- The configuration procedure dynamically detects the components that are installed.
- Fewer configuration procedures are required.
- Both silent and interactive modes of operation are available.
- Most configuration information for policies and the TWSS Administration Console is migrated for you.
- You can run multiple configuration processes. Resources created in previous runs are not automatically removed, but are potentially reused.
- There is no need to undo configuration. When EAR components are not deployed, the WebSphere resources defined to support those components will remain because they may be shared with other components. They can also be used when the components are redeployed.
- The configuration procedure supports a partitioned deployment where the Access Gateway, Service Policy Manager, and the Service Platform components are on separate clusters, and it also supports a combined deployment where all

of the components are on the same cluster. The choice is driven by the selections you make during installation and the information you provide when running the First Steps configuration script.

- The First Steps configuration script complies with appropriate accessibility requirements.

## Enhancements to the Access Gateway

The Access Gateway now runs co-resident with WebSphere Enterprise Service Bus, version 6.1.0.2. It also provides Performance Monitoring Infrastructure (PMI) metrics for support.The following are the Access Gateway supported metric types:

- Startup Time Metrics on a per service context
- Successful and failed incoming request counts
- Authorized or denied request counts

**Note:** If you are running WebSphere Application Server version 6.1.0.21, the TWSS Access Gateway can coexist in the same cluster with the Service Platform components. However, when the Service Platform components are deployed in a cluster running WebSphere Application Server version 7.0.0.1, the Access Gateway must be deployed in a separate cluster.

The Access Gateway mediation primitives are enhanced as follows:

- In conjunction with the Address Masking component Web service, a new Address Masking mediation primitive is introduced.
- The SLA Enforcement mediation primitive is enhanced to track the SLA data for longer durations. You can now use this mediation primitive to maintain the traffic statistics over a significantly longer period of time than was possible in previous releases of TWSS. Additionally, all SLA enforcement is now done at the cluster level rather than at both the cluster and local levels.
- The Message Interceptor mediation primitive is used instead of the Message Logger mediation primitive. This mediation primitive has more features than the WebSphere ESB Message Logger mediation primitive. The Message Interceptor mediation primitive can capture message contents, target information in specific cases, and supports the interception of Service Message Objects (SMO). This is done by providing a target level logging capability. This mechanism allows for target address information contained in the actual messages which are logged, for tracking activity.
- The TransactionRecorder and NetworkStatistics mediation primitives are a set of inclusive default mediation primitives. They are independent of one another and have specific unique functions. The TransactionRecorder mediation primitive records all of your transaction information. Similarly, the NetworkStatistics mediation primitive records statistics for a particular service operation, such as the event type, transaction identifier, and timestamp.

For improved performance, data is now written asynchronously to the mediation primitive database rather than being written for every transaction.

## Enhancements to the Service Platform components

The Usage Record component Web service has been enhanced for improved performance, in terms of writing usage data to persistence data storage.

## Enhancements to the Service Policy Manager

In Telecom Web Services Server version 7.0, a combined deployment which enables the Service Policy Manager, Access Gateway and Service Platform components to coexist within the same cluster. However, you may want a separate cluster from a deployment perspective.

## Enhancements to the Parlay Connector

To ease installation and deployment, all of the Web service implementations including those that operate over Parlay 3.x and 4.x APIs, are packaged in separate and distinct enterprise application (EAR) files.

The Parlay Connector is packaged in a separate EAR file. This allows for a separate deployment and management, database access performance improvements, and the reduction of object storage and ORB calls. The Parlay Connector follows a remote communication model. A remote interface has been exposed to the applications to obtain the Parlay connector services.

## Presence supplier PUBLISH operation

Presence PUBLISH is a client-generated Web service request that creates an event state associated with the address of record (AOR). This Web service enables you to publish presence information by a presentity or by an entity whose presence information is of interest to you.

The following are examples of presence attributes: current activities, environments, communication means, and contact addresses.

## New Web service implementations

Telecom Web Services Server version 7.0 includes support for the following new Web service implementations:

**Parlay X Call Handling over Parlay**

The Parlay X Call Handling over Parlay component establishes routing and call handling rules that the service executes when a call initiation activity is detected at the gateway. It provides a mechanism for applications to specify how the calls are to be handled for a specific phone number using the Parlay X 2.1 Web service interface. The component processes call events which arrive from network elements representing event notification data, and handles the call instance appropriately according to the defined rules.

The following actions are commonly utilized:

- Call accepting: Accepts calls only from numbers in a list.
- Call blocking: Blocks calls if they are on a blocking list.
- Conditional call forwarding: Changes the destination of a call to another number for a specific calling number.
- Unconditional call forwarding: Changes the destination of a call to another number.

The following operations are supported:

- setRules
- getRules
- clearRules

- setRulesForGroup

**Parlay X Call Notification over Parlay**

The Parlay X Call Notification over Parlay component sends Web service requests to the network mapper for creating and registering notification information using the Notification Management component Web service, and the reporting of network related events to the PX Notification component Web service. The Call Notification component provides the Parlay X 2.1 based Web Service that accepts requests for Call Notification Manager and Call Direction Notification Manager, while controlling various checks which are done to ensure that the requests are valid. The parlay connector is used to access the MPCC service manager and forwards the request to the gateway based on the Parlay 4.2 Multi-Party Call Control specification. This is utilized for downstream processing. Also, the Call Notification component processes information that arrives from network elements that represent event notification data, and then forwards the notification data to the appropriate registered subscribers.

The following operations are supported:

- startCallNotification
- stopCallNotification
- startCallDirectionNotification
- stopCallDirectionNotification

**Parlay X Third Party Call over Parlay**

The Parlay X Third Party Call over Parlay component accepts and verifies third-party call requests such as initiating a call session, terminating a call session, and retrieving call status information. It provides the Parlay X 2.1 based Web Service which accepts and controls various checks which validate the requests, and also maps the requests to the Parlay Connector based on Multi-Party Call Control specifications for downstream processing.

The following operations are supported:

- makeCall
- getCallInformation
- cancelCall
- endCall

**Address Masking component Web service**

The Address Masking component Web service offers masking, unmasking, shadowing, MaskingWithExpiry, UnmaskingWithExpiry, and unshadowing capabilities, which uniquely identifies a subscriber’s identity (sip or MSISDN address). This common component Web service provides the functionality for masking and shadowing addresses, and also unmasking and unshadowing them back to their original values.

**Note:** The Address Masking component Web service, is deployed as a Service Platform components, and uses the SPM for masking configurations.

## Enhancements to existing Web service implementations

The WAP Push over SMPP Web service implementation use predefined values or ESMC settings. You can now use the TWSS Administration Console to modify these values.

The Parlay X Terminal Location over MLP Web service implementation is enhanced to support various shapes for representing geographical areas in which a mobile subscriber can be located. These shapes are defined in versions 3.1 and 3.2 of the MLP protocol.

**Note:** Previously, only the CircularArea shape was supported.

In addition, the following Web service implementations are enhanced to provide improved performance:

- WAP Push over SMPP
  - Accepts requests for sending WAP Push messages
  - Controls various checks that are done to ensure the requests are valid
  - Forwards the requests to the appropriate network elements for downstream processing
- Parlay X SMS over SMPP
  - Synchronous messaging
  - Messaging without status maintenance
  - Default endpoint for orphan MO (mobile originated) messages
- Parlay X Multimedia Messaging over MM7
  - Processes information that arrives from network elements representing confirmation of delivery notification and event notification
  - Forwards the notifications to the appropriate registered subscribers

  **Note:** For better performance, you can use the HTTP connector instead of the MM7 JCA adapter. It is used to manage connections between the service implementation and the network entity. The short code will be used as an example of the address data item.

### Support for new version of WebSphere Application Server

The IBM WebSphere Telecom Web Services Server product now runs with the following WebSphere products:

- WebSphere Application Server Network Deployment version 7.0.0.1 or version 6.1.0.21
- WebSphere Enterprise Service Bus version 6.1.0.2

For information to help you select the best combination of software products to run in your IBM WebSphere Telecom Web Services Server environment, refer to the topics in the *Planning* section of this information center.

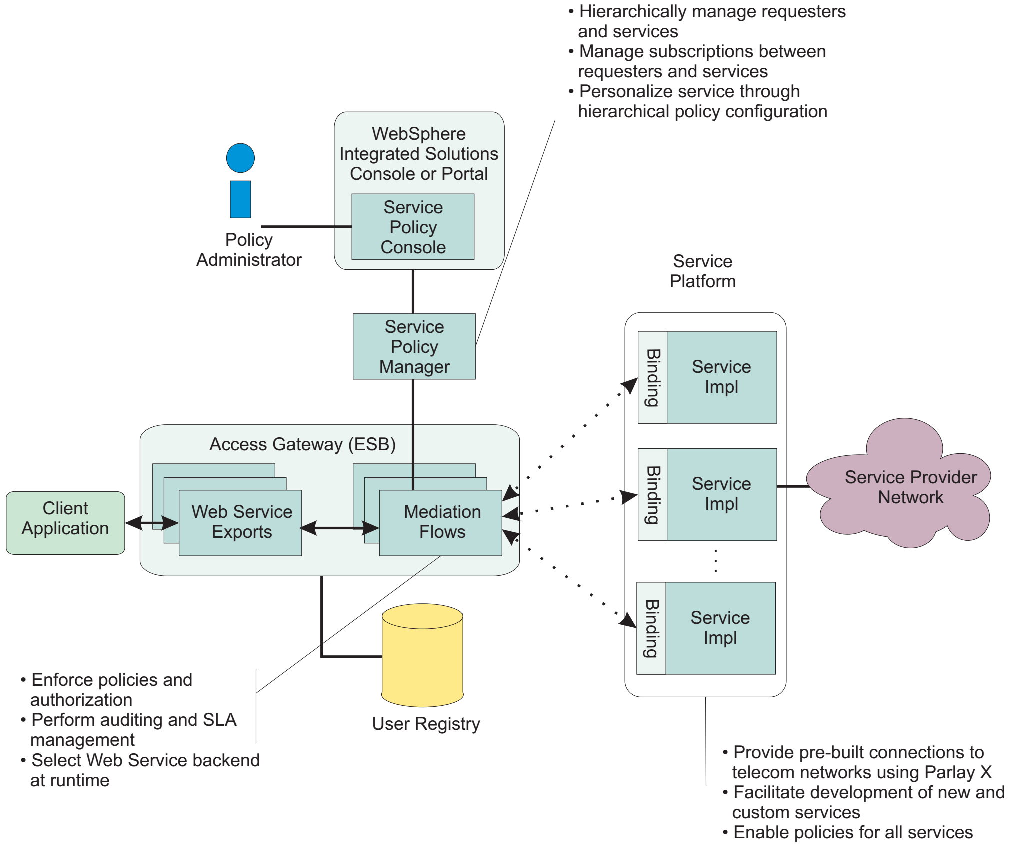
## Telecom Web Services Access Gateway

The Telecom Web Services Access Gateway provides policy-driven traffic monitoring, message capture, authorization, and management capabilities. These services are provided at the application layer, and is enforced for each Web service request using knowledge of the requester, target service, and invoked operation.

The Access Gateway provides flexibility for the construction of tailored message processing logic, in accordance with your enterprise’s network policies. This logic can be customized to meet your network requirements.

All Web service requests and responses pass through and are inspected by the Access Gateway. It acts as an intermediary for all communications between client and service endpoints by processing both incoming and outgoing requests. Mediation primitives within the Access Gateway have access to the SOAP message in its entirety in the form of a Service Message Object (SMO), including headers and body. This allows mediation primitives to determine the initiating requester, target service, and invoked operation. In addition, policy data is passed between mediation primitives in the form of additional XML elements within the SOAP header.

The Access Gateway extends WebSphere ESB by adding policy-driven processing elements and new components to monitor Web service traffic. Using WID tooling, you can model your network policies, enable new features, and deploy customized logic into the Access Gateway.


• Hierarchically manage requesters and services
• Manage subscriptions between requesters and services
• Personalize service through hierarchical policy configuration
WebSphere Integrated Solutions Console or Portal
Service Policy Console
Policy Administrator
Service Policy Manager
Service Platform
Binding
Service Impl
Binding
Service Impl
Binding
Service Impl
Service Provider Network
Access Gateway (ESB)
Client Application
Web Service Exports
Mediation Flows
User Registry
• Enforce policies and authorization
• Perform auditing and SLA management
• Select Web Service backend at runtime
• Provide pre-built connections to telecom networks using Parlay X
• Facilitate development of new and custom services
• Enable policies for all services

## Enhancements for version 7.0

The Access Gateway now runs co-resident with WebSphere Enterprise Service Bus, version 6.1.0.2. It also provides Performance Monitoring Infrastructure (PMI) metrics for support.The following are the Access Gateway supported metric types:

- Startup Time Metrics on a per service context
- Successful and failed incoming request counts
- Authorized or denied request counts

**Note:** If you are running WebSphere Application Server version 6.1.0.21, the TWSS Access Gateway can coexist in the same cluster with the Service Platform components. However, when the Service Platform components are deployed in a cluster running WebSphere Application Server version 7.0.0.1, the Access Gateway must be deployed in a separate cluster.

The Access Gateway mediation primitives are enhanced as follows:

- In conjunction with the Address Masking component Web service, a new Address Masking mediation primitive is introduced.
- The SLA Enforcement mediation primitive is enhanced to track the SLA data for longer durations. You can now use this mediation primitive to maintain the traffic statistics over a significantly longer period of time than was possible in previous releases of TWSS. Additionally, all SLA enforcement is now done at the cluster level rather than at both the cluster and local levels.
- The Message Interceptor mediation primitive is used instead of the Message Logger mediation primitive. This mediation primitive has more features than the WebSphere ESB Message Logger mediation primitive. The Message Interceptor mediation primitive can capture message contents, target information in specific cases, and supports the interception of Service Message Objects (SMO). This is done by providing a target level logging capability. This mechanism allows for target address information contained in the actual messages which are logged, for tracking activity.
- The TransactionRecorder and NetworkStatistics mediation primitives are a set of inclusive default mediation primitives. They are independent of one another and have specific unique functions. The TransactionRecorder mediation primitive records all of your transaction information. Similarly, the NetworkStatistics mediation primitive records statistics for a particular service operation, such as the event type, transaction identifier, and timestamp.

## Service Policy Manager

The Service Policy Manager provides management, storage, and retrieval functions for the policy configuration data, and the runtime data used to customize service delivery for a given requester. An enterprise-level administrator can use the Service Policy Manager to manage definitions of third-party requesters, service definitions, and service relationships. Using policy management capabilities, administrators can personalize the services that are provided to groups and to individual requesters in a way that is scalable.

### Structure and function of the Service Policy Manager

The Service Policy Manager is split into two parts: the runtime subcomponent and the console.

The runtime subcomponent provides the actual storage, management, and policy resolution capabilities. Its functions can be accessed through two sets of interfaces, both of which are available either through Web services or through Jython scripting (using the WebSphere Application Server wsadmin tool), or through MBeans that can be accessed within wsadmin. The wsadmin tool now supports the Jython scripting for advanced automation capabilities.

- The policy access interface is used by non-administrative applications to lookup policies.
- A set of administrative interfaces through which requesters, services, subscriptions, data types, and policy values can be managed

The Service Policy Manager resolves policy values using a hierarchical algorithm. The requester and service information can be organized into a tree hierarchy that allows for groupings of requesters and services. The subscriptions can be set within the requester tree scope, and policies can be set within the requester and service tree scopes. The lookup process is hierarchical, allowing requesters and services that are lower in the tree to inherit subscriptions and policy values from their parents.

Each service represents an interface that is managed within the Service Policy Manager. Multiple service implementations, or backend implementations, can be registered within the Service Policy Manager, and policies can be administered across the service as a whole or within the context of each unique backend.

A special value of ALL can be used to apply the policy across one element of the scoping. For example, defining a policy at a scope of (**ALL, myservice, ALL**), will define an attribute or value pair for the service myservice across all requesters and operations. The Service Policy Manager uses a hierarchical resolution algorithm to determine the final set of policy attribute or value pairs for a given service context.

The Service Policy Manager console provides a graphical Web interface for using the runtime subcomponent to manage various entities. You can deploy the console either in standalone fashion or within a full portal runtime environment.

## Benefits of the Service Policy Manager

The Service Policy Manager component provides the following benefits:

- **Centralized repository and management capability**: The Service Policy Manager is intended to be the focal point for managing runtime configuration for all TWSS Web service implementations. Service policy values can be updated dynamically, taking effect for subsequent message processing. In a large scale deployment, a single Service Policy Manager cluster can be shared by multiple Access Gateway clusters. When a centralized policy management system already exists, policy data can be pushed to the Service Policy Manager system through integration with Web services.
- **Grouping requester policies and subscriptions**: An arbitrary requester hierarchy is supported for managing requesters. This hierarchy allows the administrator to create subscriptions and set policies across groups of requesters by creating these entities higher up in the hierarchy. This allows use cases where all members of a *gold* class of service can be subscribed to and inherited by all members of the requester group. Requester groups are internal administration constructs only and are not accessible to fetchers of policy information. The requester tree can be organized according your needs.
- **Grouping services**: An arbitrary service hierarchy is supported for managing services. This hierarchy allows administrators to set policy values at shared scopes across all or a subset of services. Service groups are internal administration constructs only and are not accessible to fetchers of policy information. The service tree can be organized according to your own management structure.
- **Associating multiple backend implementations with an individual service**: The Service Policy Manager allows for registering multiple backend implementations for a given service that may be deployed in the environment simultaneously. The exact backend implementation to use for each request is chosen based on the requester subscription.
- **Hierarchical resolution of policies**: The Service Policy Manager supports setting policy information at three different scopes: requester, service (including backend implementations), and operation. To resolve policies, the Service Policy Manager uses a hierarchical algorithm based on the requester and service tree, which combines subscriptions and policy scopes by having entities lower in the tree overriding ones higher in the tree. The intention of this capability is to support

management of large sets of requesters and services, while enabling evolution of the set of deployed services within the environment.

## Service Platform components

The Service Platform components provide a common service implementation function when deployed on the same level of the WebSphere Application Server platform. This enables more efficient and smaller deployment platform sharing. The Service Platform components is reusable and can be utilized to support custom built services.

### The Common components

All service implementations share a set of the Service Platform components, to facilitate the rapid development of new services. Additionally, the Telecom Web Services Server supports integration with the external privacy management system, or the privacy service.

**Note:** The recommended method of deploying the Service Platform components, is to use the local Web service invocations and the WebSphere Application Serverin-process optimization. This applies only when the Service platform common components are deployed on the same machine or node as the Service platform implementation.

The following common components are utilized by the Access Gateway and Service Platform.

- Admission Control
- Traffic Shaping
- Network Resources
- Notification Manager
- PX Notification
- Address Masking
- Fault and Alarm
- Usage Record
- Privacy Web Service

## Admission Control component Web service

The Admission Control component Web service establishes a configurable threshold for the number of requests from Web service implementations which are admitted into the system (server and cluster level), within a given time interval.

Immediately upon receipt of a Web service request, service implementation logic invokes the Admission Control component Web service to determine whether to accept or reject the request from the Web service clients.

If the request is accepted, processing continues as normal.

If the request is rejected, the Admission Control component Web service returns an error message to the requester.

This component enables you to control how much traffic can be accepted by each service for each member of the cluster and across the cluster. Controlling traffic helps ensure that in the event of a prohibitively large number of requests, servers do not exceed planned capacity.

## Traffic Shaping component Web service

The Traffic Shaping component Web service controls the rate at which traffic can be directed toward a given network element across the cluster.

A Web service implementation can interact with one or more network element. Each network element is associated with a resource definition which is created within the TWSS Administration Console, and defines the rate of traffic that can be directed toward the network element.

## Network Resources component Web service

The Network Resources component Web service provides a way to define specifications for network resources (elements) with which the service platform will interact. Specifications that you design using the Network Resources component Web service, is used to tailor the way in which the service platform behaves toward each resource.

For example, the Service Platform can use a network element's message processing capacity to control the rate at which traffic is generated towards the element. Other specifications can be used for storing additional properties regarding protocol interactions with the element.

This component provides a Web service interface to access resource specification properties from other elements. It is used by the Traffic Shaping component Web service to fetch network resource properties. Resource specifications are intended to be a flexible set of properties, and thus they can be used by other components and services.

## Notification Management component Web service

The Notification Management component Web service provides notification tracking and administrative functionality. Registered notifications are stored in a backing database and are made available for queries or terminations, through a Web service API.

Web service implementations that use Notification Management require that a client be configured using the configuration pages in the TWSS Administration Console.

**Note:** Not all of the TWSS Web service implementations require the Notification Management component Web service. Some provide a notification option that uses the Notification Management component Web service.

Notification Management also provides statistics that an administrator can use to gain insight into notification delivery in a running system. This component is intended only for capturing information about all notifications on the platform, and for capturing statistics around notification delivery attempts. Individual Web service implementations still should manage the storage of notification data in whatever manner is most efficient for them.

Notification Management is a Web service and is defined by WSDL. It can query notification information and terminate active notifications. The component provides simple information that can be used by an administrator or care representative, to view information about active notifications in the system. The API also allows those individuals to reset outstanding notifications through termination. Upon termination, the client-side application would need to reestablish the notification.

The component provides the following functionality:

**Register notifications**
A service implementation can call the Notification Management component to register notifications in the Notification Manager database.

**Complete or terminate notifications**
A service implementation can call the Notification Management component to remove notification registrations from the Notification Manager database.

**Tracking and updating delivery statistics in Notification Management from the PX Notification**
The PX Notification component Web service component, can call the Notification Management component to update delivery statistics in the Notification Manager database for tracking purposes.

**Query the Notification Management component**
An application can call the Notification Management component to query statistics, find notification records, and delete records from the Notification Manager database.

**Terminate currently processing notifications**
An application can call the Notification Management component to send a termination request to the service implementation, which creates the notification so that the service implementation can do its own tear down.

For more information about the Notification Management component Web service, refer to the topic *Planning notification* in this information center.

## PX Notification component Web service

The PX Notification component Web service provides facilities for notification delivery to the requestor endpoint directly or through the Access Gateway.

This component also provides an asynchronous model for invocation, in which an application can send a notification, continue its processing without blocking, and then receive a delivery confirmation. This feature is not configurable from the end user standpoint, and is only configurable from the Web service point of view. It supports all services having notification operations.

A retry mechanism has been added to the notification delivery of several Parlay X services that retries the endpoint for a configurable number of times, before logging a failure. The endpoint could be the ESB endpoint or the requestor's endpoint. The retry is done at the PXNotification component scope and is done as long as the endpoint URI is not accessible and the retry count is not exhausted. A retry delay is also configurable for the pause time between the retries.

**Note:** The retry does not apply to faults within the ESB notification flow. Such faults will be logged.

Three delivery mode options indicate how the notification needs to be conveyed to the requestor endpoint. This is usually set by the Web service implementations and can be overridden at the PX Notification component level.

The following are the delivery mode options:

- **Inline**: A single thread of execution from the service implementation to the endpoint (ESB or requestor).
- **Asynchronous**: The service implementation hands over the request to the PX Notification component, which in turn places the request in a JMS queue. The service implementation is not blocked for a response from the endpoint. Delivery confirmation can be obtained using the Delivery Confirmation Callback interface.
- **Mode set by the service implementation**: This is the default option. The delivery mode is usually set by the Web service implementation and can be overridden at the PX Notification component level. Therefore, whatever delivery mode (Inline or Asynchronous) is chosen by the service implementation is used by the PX Notification component.

*Table 1. Delivery modes used by the services*

| Parlay X service implementations | Delivery mode used for notification delivery through PX Notification |
|---|---|
| Terminal Location/Parlay | Inline |
| Terminal Status/Parlay | Asynchronous |
| Call Notification/Parlay | Inline |
| SMS/Parlay and SMS/SMPP | Inline |
| Call Notification/SIP | Asynchronous |
| Presence/SIP | Asynchronous |
| Terminal Status/SIP | Asynchronous |
| MMS/MM7 | Inline |
| Terminal Location/MLP | Inline |

For more information about the PX Notification component Web service, refer to the topic *Planning for notification* in this information center.

## Address Masking component Web service

The Address Masking component Web service provides features to hide the actual identity of a subscriber from third party applications (TPA). Third parties are internal and external customers of service providers.

**Note:** They can be different internal divisions within an organization that wish to develop new services based on core infrastructure function or external customers that wish to integrate new or enhance existing services.

Address masking is provided by the Address Masking mediation primitive (part of the processing flow for the Access Gateway), working in conjunction with the Address Masking component Web service (a Service Platform component).

You can use address masking to mask, unmask, shadow, or unshadow the specific addresses that identify the subscriber (MSISDN or SIP address) to an application. The Address Masking mediation primitive extracts occurrences of any such address from the message (SMO) and sends the addresses as key-value pairs to the

Address Masking component Web service. The masking service replaces all addresses contained in the request with their masked values, or vice versa, depending on the operation being performed.

The operations being performed are configured as policies, using the Service Policy Manager. The mediation primitive looks up the policies and behaves accordingly.

The following operations are provided:

**Masking**
Encrypts the address fields contained in a request message, using an encryption algorithm. The encrypted address value is composed of numbers, alphanumeric characters and special characters.

**Unmasking**
Decrypts one or more masked address values contained in a request message. This operation and the masking operation are complementary.

**Shadowing**
Replaces the MSISDN contained in a request message with a pseudo-MSISDN value, and returns the pseudo value to the requester. For example, the MSISDN 9818010846 might be replaced by 98180XXXXX where X is any digit from 0 to 9. The number of digits to be shadowed is user configurable.

**Note:** Pay attention to the fact that the algorithm used for shadowing is very difficult to decipher, but not impossible. Also, be aware that there is a possibility of the generated shadowed number appearing as a valid MSISDN number. Hence, Shadowing is a comparatively vulnerable option as compared to the Masking or MaskingWithExpiry operations. If you are concerned about security, you should not use the Shadowing operation.

**Unshadowing**
Replaces the pseudo-MSISDN value contained in a request message with the actual MSISDN, and returns the unshadowed MSISDN to the requester. This operation and the masking operation are complementary.

**MaskingWithExpiry**
Replaces the address fields with a pseudo (random) number, which is configured for expiry–in other words, the pseudo number can be used only for a certain period of time, after which the address expires. Any subsequent requests that use an expired number are rejected. This operation and the UnmaskingWithExpiry operation are complementary.

**UnmaskingWithExpiry**
Given a masked number, it retrieves the corresponding original number from a database and sends the response back to the user. This operation and the MaskingWithExpiry operation are complementary.

**Note:** The Address Masking component Web service mediation primitive is deployed as a Service Platform components, and the mediation primitive uses the Service Policy Manager for masking configurations. This mediation primitive is available only with the version 7.0 levels of the TWSS Access Gateway and Service Policy Manager.

## Fault and Alarm component Web service

The Fault and Alarm component Web service records information about reoccurring faults and notifies the administrator with alarms. The alarms can be either JMX notifications or common base events.

When encountering error conditions, Web service implementations generate fault information and potentially emit alarms for severe error conditions. They emit fault and alarm information as common base events, using CEI and JMX management notifications. The common base event is an extensible XML schema for describing event information.

CEI is used as the infrastructure for notifying interested parties of events, and it provides a persistent event repository that supports flexible event queries based on XPath. This component also includes an MBean that generates JMX notifications directly. The use of CEI enables integration with other WebSphere products, such as Tivoli® security products.

## Usage Record component Web service

The Usage Record component Web service records information about service usage and fault information. It stores usage records generated by the Web service implementations deployed in your network.

Each Web service implementation generates service usage records that describe how the service was used, for accounting and billing purposes. Usage records are stored in a database table for processing by a billing mediator application. A usage record typically corresponds to a single row in the table, although large usage records are broken into multiple rows.

The attributes and values stored within usage records are unique to each service implementation.

## Integrating with a Privacy Web Service

WebSphere Telecom Web Services Server supports integration with an external privacy management system, also referred to as a Privacy Web Service. Integration can be configured for each Web service implementation. To integrate, an appropriate and valid Web service endpoint must be configured for the Web service implementations that need to use it.

Privacy can be evaluated for each requester, service, operation, and target. For security reasons, each Web service implementation is configured with a default local Web service endpoint that will cause a privacy error when processing requests. This endpoint should be modified with the correct endpoint of the Privacy Web Service, or if you are not using a Privacy Web Service, this endpoint should be disabled by setting the configuration field to a blank value.

# Web service implementations

The Telecom Web Services Server product supports a large number of Web service implementations, many of which are based on Parlay X 2.1.

The Web service implementations are listed according to the protocols they use to interact with the Network Entity.

## Web service implementations based on SIP/IMS

Telecom Web Services Server supports Web service implementations that operate over SIP/IMS.

### Parlay X Call Notification over SIP/IMS: Introduction

Parlay X Call Notification over SIP/IMS allows third-party client applications retrieval of event information from the network. A third-party application can receive information about the call event through a notification registration.

The following types of events can be reported:

- Called party is on the phone
- Called party does not answer the phone
- Called party is unreachable
- A call was attempted

**Note:** This Web service implementation does not actually process or handle a call. Instead, it notifies a registered party of a call event or of an interested called party. This registered party is an application that is registered to be notified whenever there is a call event for the interested called party.

A NotificationAdministrationSupport interface is also available. This operation issues a `TerminateNotificationRequest` message, which is designed to terminate notifications by removing the NOTIFICATION table entry. This is based on the results rendered from the TerminateNotifications request.

### Parlay X Presence over SIP/IMS: Introduction

Parlay X Presence over SIP/IMS enables client applications to publish the presentity's presence information to the presence server, and retrieves the presentity presence information from the presence server-either by looking up information or asking for notifications.

By integrating with this Web service implementation, your client applications can use Web Services to subscribe to a presentity, synchronously query the current presence information for a presentity, receive asynchronous notifications about changes in the presence information for a presentity, publish the presence information of a presentity, and unsubscribe from a presentity.

### Parlay X Terminal Status over SIP/IMS: Introduction

Parlay X Terminal Status over SIP/IMS allows the client application to request the status of an IMS terminal that work in conjunction with the IBM WebSphere Presence Server Component.

Parlay X Terminal Status over SIP/IMS queries Presence Server to retrieve the status of the IMS terminal (or terminals). When the state of the terminal changes, Presence Server sends a notification to the Parlay X Terminal Status over SIP/IMS Web service.

### Parlay X Call Handling over SIP/IMS: Introduction

The Parlay X Call Notification over SIP/IMS component provides the Parlay X 2.1-based Web service, which accepts requests for call handling services, controls various checks that are done to ensure the requests are valid, and maps the requests to the parlay connector. This feature provides operations to set rules for a network terminal or a group and allows it to accept, block, or route calls to another terminal. It also provides the functionality to view or clear a pre-existing rule.

### Parlay X Third Party Call over SIP/IMS: Introduction

Parlay X Third Party Call over SIP/IMS enables third-party client applications to make calls and get call status information, which provide the ability to initiate a call from a network entity between two completely different users or user agents.

## Web service implementations based on IBM WebSphere software

Telecom Web Services Server (TWSS) supports Web service implementations that interoperate with other components within the IBM WebSphere product group–including WebSphere Application Server, IBM XDMS, and other instances of TWSS.

### Parlay X Payment (PostPaid): Introduction

Parlay X Payment (PostPaid) enables third-party client applications to send charging information, by enabling the application to charge an amount against a user account or place money into a user account. It interacts with the Usage Record Web Service and writes data to the Usage Record database table.

### Parlay X Address List Manager over XCAP: Introduction

Parlay X Address List Manager over XCAP provides Web service-based mechanisms to create, read, update, and delete group definitions stored within the group list manager: IBM XDMS.

This service interconnects with IBM XDMS, through which it accesses to the XCAP protocol. This service is designed to work with any XCAP server; however, integration is ensured with IBM XDMS.

## Web service implementations based on Direct Connect protocols

Telecom Web Services Server supports Web service implementations that operate over Direct Connect protocols such as SMPP, MLP, and MM7.

### WAP Push over SMPP

WAP Push over SMPP sends SMS messages to the handset as a means of pushing data (for example, a URL link), to the handset.

WAP Push over SMPP forwards the message to the SMPP connector for processing, and then sends it to the SMSC.

### Parlay X SMS over SMPP

Parlay X SMS over SMPP provides operations for sending an SMS message to the network, monitoring the delivery status of a sent SMS message, asynchronously receiving notification of message delivery status, notification functionality, and receiving MO messages.

Parlay X SMS over SMPP also supports requests to send an SMS logo to a specified address or set of addresses (sendSmsLogo) and requests to send an SMS ring tone (sendSMSRingtone). It interacts with the native SMPP network using a JCA adapter.

### Parlay X Terminal Location over MLP

Parlay X Terminal Location over MLP enables applications to send Web service requests to TWSS requesting the Terminal Location services defined by the Parlay X 2.1 specification and register for Terminal Location notifications, and terminal distance functionality.

This service implementation offers a simple Web service API that adapts the Parlay X 2.1 API to the MLP 3.1 and 3.2 specifications.

### Parlay X Multimedia Messaging over MM7

Parlay X Multimedia Messaging over MM7 accepts requests for MMS services and performs various checks to ensure that the requests are valid.

## Web service implementations based on Parlay

Telecom Web Services Server supports Web service implementations that operate over Parlay 3.x and 4.x APIs with Parlay 5.0 Multimedia Messaging (MMM).

These service implementations interact with a Parlay gateway using a Parlay Connector.

### Parlay X Terminal Status over Parlay: Introduction

Parlay X Terminal Status over Parlay enables applications to send status and notification requests for a terminal, as defined by the Parlay X 2.1 specification. This service also supports group level operations and sends notifications to the Parlay X Terminal Status over Parlay Web service, when the status of the terminal changes.

### Parlay X Terminal Location over Parlay: Introduction

Parlay X Terminal Location over Parlay provides operations for sending Terminal Location requests and registering for Terminal Location notifications.

### Parlay X SMS over Parlay: Introduction

Parlay X SMS over Parlay provides operations for sending an SMS message to the network, monitoring the delivery status of a sent SMS message, and asynchronously receiving notification of message delivery status.

### Parlay X Call Handling over Parlay: Introduction

Parlay X Call Handling over Parlay component provides the Parlay X 2.1 based Web service which accepts requests for call handling services, controls various checks that are done to ensure the requests are valid, and maps the requests to the parlay connector. This feature provides operations to set rules for a network terminal or a group and allows it to accept, block, or route calls to another terminal. It also provides the functionality to view or clear a pre existing rule.

### Parlay X Call Notification over Parlay: Introduction

Parlay X Call Notification over Parlay processes information that arrives from network elements representing event notification data. It enables client applications to create notifications, register notification information, and report network related events to the PX Notification component Web service.

This Web service implementation accepts requests for the Call Notification manager and the Call Direction Notification manager, and performs various checks to ensure that the requests are valid. The requests are then forwarded to the Parlay Connector, based on the Parlay 4.2 Multi-Party Call Control specification for downstream processing.

This Web service implementation also provides administrative support to terminate a registered notification.

## Parlay X Third Party Call over Parlay: Introduction

Parlay X Third Party Call over Parlay enables third-party client applications to make calls, end calls, cancel calls, and get call status information.

# Chapter 2. Planning to install WebSphere Telecom Web Services Server

WebSphere Telecom Web Services Server includes the Access Gateway, theService Policy Manager, a set of Service Platform components, and a number of Web service implementations. All of these can be installed across multiple servers in the network.

When planning to install Telecom Web Services Server, you must consider how you want to deploy the components in your network.

Decide which set of Web service implementations you want deployed in your network. The Service Platform components are deployed locally with the Web service implementations, and are distributed among each Service Platform application server node in the cluster.

The Service Policy Manager can be installed in the same cluster with the Service Platform components and the Web service implementations, or it can be installed in a separate cluster of its own. The Service Policy Manager console is normally deployed within the same cluster as its runtime component.

Regardless of the desired network configuration, the components should be installed into a clustered environment with a deployment manager and a managed cell. A common configuration has two clusters: an Access Gateway cluster and a Service Platform cluster. From the vantage point of WebSphere Application Server, the use of separate clusters means that each node is treated as a separate managed entity.

Each component requires WebSphere Application Server. The Access Gateway requires WebSphere ESB, which includes WebSphere Application Server.
- The Service Platform components deploy on WebSphere Application Server version 7.0.0.1 or 6.1.0.21.
- The Access Gateway deploys on WebSphere ESB version 6.1.0.2 (which includes WebSphere Application Server).
- The Service Policy Manager deploys on WebSphere Application Server version 7.0.0.1 or 6.1.0.21.

For a complete list of software requirements, including supported application server and database versions, refer to the topic *Hardware and software requirements*.

## Hardware and software requirements

Specific hardware and software is required before you can begin the installation process.

### Hardware requirements

Hardware requirements vary, depending on the operating system on which you plan to deploy the IBM WebSphere software for Telecom products.

Before you begin the installation, one of the following operating-system platforms must be installed and configured. Choose a platform to display a detailed list of hardware requirements.

“AIX”
“Linux on PowerPC”
“Linux on Intel”
“Solaris on Intel” on page 21
“Solaris on SPARC” on page 21

This information represents the minimum requirements. For greater performance and scalability, additional hardware may be needed.

## AIX®

**Processor**
Power 5, 1.5 GHz, 32- and 64-bit

**Physical memory**
4 GB minimum, 2 GB per JVM recommended

**Disk space**
2 GB of free space (minimal)

6 GB of free space recommended

**Other:** CD-ROM or access to shared network drive where CD images are available

## Linux® on PowerPC®

**Processor**
Power 5, 1.5 GHz, 32- and 64-bit

**L2 cache**
L2 cache for 2.8 GHz processor must be 512 KB

L2 cache for 3.4 GHz processor must be 1 M

**Physical memory**
4 GB minimum, 2 GB per JVM recommended

**Disk space**
2 GB of free space (minimal)

6 GB of free space recommended

## Linux on Intel®

The following configuration is supported for Intel x86 platforms:

**Processor**
Pentium® 4, a minimum of 2 processors is required

2.8 GHz (32- and 64-bit)

**L2 cache**
L2 cache for 2.8 GHz processor must be 512 KB

L2 cache for 3.4 GHz processor must be 1 M

**Physical memory**
4 GB minimum, 2 GB per JVM recommended

**Disk space**
2 GB of free space (minimal)

6 GB of free space recommended

**Other** CD-ROM or access to shared network drive where CD images are available

Hyper-threading should be enabled

## Solaris on Intel

The following configuration is supported for Intel x86 platforms:

**Processor**
Model R2A-8853: 2 CPU 3.3GHz

Model 56A-4364: 2 CPU 2.13GHz

**L1 cache**
Model R2A-8853:
  Installed size: 32 KB
  Maximum installed size: 32 KB

Model 56A-4364:
  Installed size: 32 KB
  Maximum installed size: 64 KB

**L2 cache**
Model R2A-8853:
  Installed size: 6 MB
  Maximum installed size: 6 MB

Model 56A-4364:
  Installed size: 2 MB
  Maximum installed size: 8 MB

**Physical memory**
Model R2A-8853: 4 GB

Model 56A-4364: 8 GB

**Disk space**
2 GB of free space (minimal)

4 GB of free space recommended

**Other** CD-ROM or access to shared network drive where CD images are available

## Solaris on SPARC

The following configuration is supported for UltraSPARC-T1 platforms:

**Processor**
4 virtual 1400 MHz

**L1 and L2 cache**
16 KB primary instruction cache per core:
  Parity protected and single-bit error recoverable
  4-way set-associative

8 KB primary data cache per core:
    Parity protected and single-bit error recoverable
    4-way set-associative

3 MB unified level 2 cache:
    12-way set associative, 4 banks
    ECC

**Physical memory**
5120 MB

**Disk space**
2 GB of free space (minimal)

4 GB of free space recommended

**Other** CD-ROM or access to shared network drive where CD images are available

## Software requirements

Required software includes the operating system, the WebSphere Application Server Network Deployment product (also referred to as WebSphere Application Server), Java, and a database component. TWSS also requires WebSphere ESB and a Web browser.

The information provided here is intended for a basic installation that is not scaled or fully deployed.

**Note:** WebSphere Integration Developer (WID) version 6.1.0.102 is used to customize the Access Gateway default flows.

The following software should be installed and configured before you begin the installation process:

"Operating systems"
"Application servers"
"WebSphere Enterprise Service Bus" on page 23
"Browsers" on page 23
"Java version" on page 23
"Databases" on page 23

### Operating systems

The following operating systems are supported:

Linux Red Hat Enterprise Linux AS 5.0 Update 2
Linux SUSE Linux Enterprise Server 10 SP1
AIX AIX 5L 5.3 TL 07 04-0818
Sun Solaris 10 (64-bit on Intel or 32-bit on Sparc)

### Application servers

One of the following application server offerings is required:

- WebSphere Application Server Network Deployment, version 7.0.0.1

**Note:** Interim fix 76513 for WebSphere version 7.0.0.1 is required if you plan to use the Notification Management or PX Notification Web services.

- WebSphere Application Server Network Deployment, version 6.1.0.21

For a list of required WebSphere Application Server fixes, refer to the readme file, `WebSphereSoftwareForTelecomReadme.html`, on the QuickStart CD.

## WebSphere Enterprise Service Bus

The Access Gateway requires WebSphere Enterprise Service Bus (WebSphere ESB). WebSphere ESB requires a different version of WebSphere Application Server Network Deployment than the rest of TWSS. Before attempting to install the Access Gateway, you must install the following software:

- WebSphere Enterprise Service Bus, version 6.1.0.2 (which includes the appropriate version of WebSphere Application Server Network Deployment)
- Web Services feature pack for WebSphere Application Server Network Deployment
- JDK, version 1.5.0 SR 8

**Important:** Do not attempt to install WebSphere Enterprise Service Bus to a directory path that includes National Language Characters. If the path contains National Language Characters, the installation will not complete.

## Browsers

One of the following Web browsers is required:

- Firefox version 2.0 or higher

  **Note:** If you are using the Service Policy Manager console with the Firefox browser, Firefox version 3.0 is strongly recommended.
- Internet Explorer version 7.0 or higher

## Java™ version

The following JDK versions are required:

- WebSphere Application Server version 7.0.0.1 requires JDK version 1.6.0 SR 3.
- WebSphere Application Server version 6.1.0.21 requires JDK version 1.5.0 SR 8.

**Note:** Your installation of WebSphere Application Server includes the correct JDK.

## Databases

Each component has different database needs. Refer to the planning section for each component to understand the database needs for that component. WebSphere Telecom Web Services Server requires multiple database tables. Scripts are provided to assist in the database table creation; however, these scripts are provided for AIX and Linux only.

The following databases are supported:

DB2 IBM DB2® Enterprise Server Edition, version 9.5 FixPak 1

Oracle Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7

## Evaluating your hardware environment

WebSphere Telecom Web Services Server installs and runs as an application on WebSphere Application Server. It can be deployed on various hardware configurations.

Before starting the installation process, review the planning information to develop a deployment strategy that will meet your network needs.

**Note:** If you are running WebSphere Application Server version 6.1.0.21, the TWSS Access Gateway can coexist in the same cluster with the Service Platform components. However, when the Service Platform components are deployed in a cluster running WebSphere Application Server version 7.0.0.1, the Access Gateway must be deployed in a separate cluster.

WebSphere Application Server supports numerous deployment topologies. It is beyond the scope of this documentation to provide detailed steps for each topology. Therefore deployment information has been grouped into a number of broad categories. Throughout the documentation the categories are used to provide a reference point. Each component has a unique deployment strategy. Prior to deployment, review all of the planning and installation information.

Here is a list of the most commonly used topologies in a WebSphere Application Server environment:

**Note:** The single server topology can be used for development or the proof of concept. The Access Gateway, Service Policy Manager, Service Platform components, and Web service implementations can be installed on a single server. However, this is not recommended for a deployment.

**Distributed topology**

The components (for example, the Web server, application server, and databases) are physically separated onto different servers. This type of topology might be used to verify functionality in an early phase of a deployment.

If you use only one server for the Access Gateway or Service Platform components, it will limit failover redundancy.

**Vertical scaling topology**

Members of a cluster exist on the same physical machine. Some services perform better with a small or moderate size Java heap. This may not utilize all of the resources of a powerful machine, so a vertically scaled deployment allows the processor and memory to be more fully utilized, while each instance can run more efficiently in a smaller JVM heap.

Frequently, vertical scaling is combined with horizontal scaling to allow both the efficient use of resources and the benefits of physical redundancy.

**Horizontal scaling topology**

Members of a cluster exist on multiple physical machines, effectively and efficiently distributing the workload of a single instance. HTTP redirector products can also be used to implement horizontal scaling. Clustering is most effective in environments that use horizontal scaling because of the ability to build in redundancy and failover, to easily add new horizontal cluster members to increase capacity, and to improve scalability by adding heterogeneous systems into the cluster.

You can combine vertical and horizontal scaling techniques to increase efficiency in the environment.

Each Web service implementation can have its own cluster of service platform instances (WebSphere Application Server with service platform common components installed locally). While multiple service implementations can be deployed on a single service platform instance, this makes it easier to size the hardware and software configurations for each Web service implementation based on their performance capabilities.

The database is shared and clustered.

**Development topology**

The IBM WebSphere Telecom Toolkit is intended for developing client applications that will remotely reference the services deployed in theWebSphere Telecom Web Services Server.

You can use WebSphere Integration Developer (WID) version 6.1.0.102 to customize the Access Gateway mediation flows. The WID plug-ins are provided so that mediation primitives can be imported into the WID tooling palette for customization of the message processing flows.

## Standard configurations for Telecom Web Services Server

WebSphere Telecom Web Services Server (TWSS) is deployed in a clustered environment. It consists of three logical components: Access Gateway, Service Platform, and Service Policy Manager. You can configure these components in several different ways.

### How the components interact

The Access Gateway builds on the WebSphere ESB (Enterprise Service Bus) platform and provides policy-driven traffic monitoring, capture, authorization, and management capabilities. These services are provided at the application layer and are enforced for each Web service request using the knowledge of the requester, target service, and invoked operation.

The Service Platform consists of an application server with enhanced support for telecom protocols and callable components for common service function. The service implementations can include custom Web service interface definitions and backend implementations. For the most part TWSS uses the Parlay X Web service APIs as a standard interface abstraction for telecom network functions. But the architecture is general enough to provide infrastructure services for any kind of Web service implementation.

The Service Policy Manager provides access function and management for policy configuration data, and the runtime data used to customize service delivery for a given requester.

A loose model based on SOAP messaging is used between the Access Gateway and the Service Policy Manager, enabling flexible deployment. However, service deployments typically require a front-end traffic management component to meet non-function service requirements, and the preferred deployment model is of the Access Gateway fronting service implementations. In addition, the WebSphere ESB platform provides rich integration capabilities for the TWSS components, providing a point of flexibility for how different components are connected.

## Initiating the TWSS request flow

For TWSS, the requester principal value is checked by the Trust Association Interceptor (TAI), if the TAI is deployed. The value, set by TWSS for the Service Platform, validates the originated request in the trusted network.

The following steps describe how the request flow is initiated for TWSS.

1. The Access Gateway authenticates the requester using WebSphere, and authorizes the requester through the use of policies. The authenticated requester then receives an asserted identity that is passed to all subsequent services.
2. The Access Gateway adds an asserted identity header to the request.

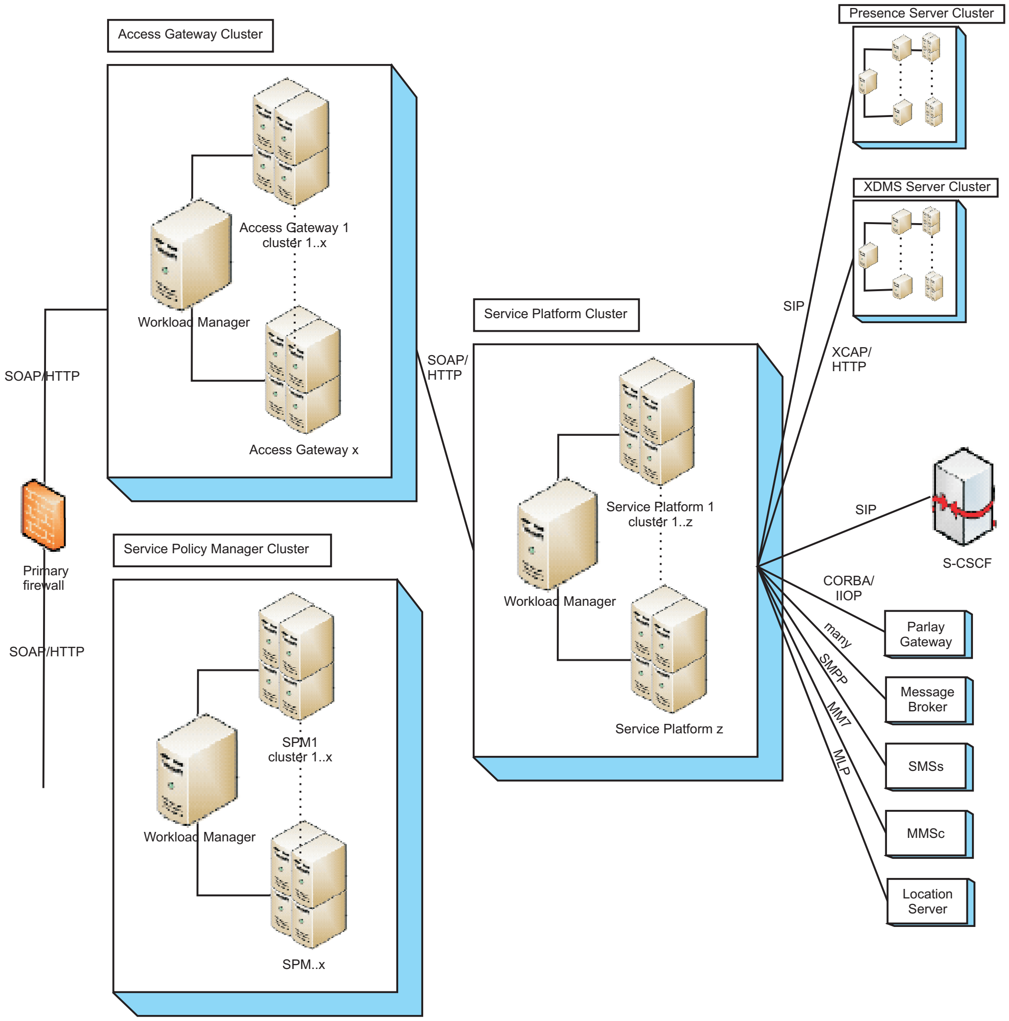
   **Note:** The Trust Association Interceptor (TAI) security component is the recommended security mechanism for handling traffic from the IMS network. A single sign-on scheme is recommended for handling transactions between the Access Gateway and the Service Platform components, and optionally between the Access Gateway and the Service Policy Manager.
3. The Access Gateway then sends the request to one of the TWSS Web service implementations to perform the work and complete the operation and the response.

## Topology

In a typical topology, there is a cluster for the Access Gateway (running with WebSphere ESB version 6.1.0.2) and another cluster for the Service Platform components (running with WebSphere Application Server version 6.1.0.x or 7.0.0.1). The Service Policy Manager can be deployed in a unique cluster or in the same cluster as the Service Platform components.

**Note:** If you are running WebSphere Application Server version 6.1.0.21, the TWSS Access Gateway can coexist in the same cluster with the Service Platform components. However, when the Service Platform components are deployed in a cluster running WebSphere Application Server version 7.0.0.1, the Access Gateway must be deployed in a separate cluster.

The following diagram illustrates a network topology in which each TWSS component is deployed in a unique cluster.



While you can define deployment managers for both the Access Gateway and the Service Platform components together on a single server, they are actually distinct deployment managers.

It is important to note that the First Steps script deploys and configures all of the base components and services that are installed on the server where it runs. Configurations in which a single server is shared between multiple clusters, and a different set of services is deployed on each cluster, are not supported.

If you are deploying Web services that are implemented over Parlay 3.x and 4.x APIs, you can also deploy the Parlay Administration Console on the same server. The Parlay Administration Console should use the same deployment manager as the Service Platform components.

## Planning to install the Access Gateway

The Access Gateway provides policy-driven traffic monitoring, capture, authorization, and management capabilities. These services are provided at the application layer, and are enforced for each Web service request using knowledge of the requester, target service, and invoked operation. The Access Gateway processes all incoming requests and messages. It is deployed on WebSphere ESB.

If you use groups, the Parlay X Address List Management Service over XCAP must be deployed. The Address List Manager service implementation enables the Access Gateway to communicate with an XCAP server, for example IBM XDMS, to resolve group members. The Group Resolution mediation primitive, part of the Access Gateway, uses the Parlay X Address List Management interface to resolve groups.

The Access Gateway is installed by the (TWSS) base installer on the deployment manager node. The First Steps script synchronizes the installation with other nodes, if any, in the cluster.

## Planning to install the Service Policy Manager

The Service Policy Manager provides a storage capability and access mechanism to enable the definition of requesters, services, and subscriptions that associate services with requesters. It can be deployed on the same cluster as the Service Platform components, or on its own cluster.

The Service Policy Manager is installed by the Telecom Web Services Server (TWSS) base installer on the deployment manager node. The First Steps script takes care of synchronizing the necessary software with other nodes, if any, in the cluster.

## Planning to install the Service Platform components

Service Platform components provide common service implementation functions when deployed on the WebSphere Application Server platform. Service Platform components are reusable and provide supporting functions for the TWSS service implementations as well as for custom Web service implementations.

The Service Platform components are installed by the Telecom Web Services Server (TWSS) base installer. The Web service implementations are installed separately, by the TWSS services installer. Both installers are intended for use only on the deployment manager node.

- WebSphere Application Server Network Deployment - Before you install and deploy the Service Platform components, the cluster should already be configured.
- Runtime components - The Service Platform runtime components are installed by the TWSS base installer, and they are intended to be installed only on the deployment manager node. The First Steps script will then synchronize the files with other nodes in the deployment.
- Database tables - The database instance is typically created before running the First Steps script, and it is not needed before running the installers.

### Admission Control component Web service

The Admission Control component Web service protects the platform on which the Service Platform components runs from overload conditions and can control the types of traffic which is admitted into the platform. The Admission Control component Web service offers two types of admission limit enforcement: locally tracked limits and cluster-wide tracked limits.

### Privacy system component

Each Web service implementation has the ability to integrate with a privacy system. When each Web service implementation is deployed, the privacy component endpoint is enabled in order to protect the Web service implementation and prevent unauthorized access.

### Traffic Shaping component Web service

The Traffic Shaping component Web service controls the flow of outbound traffic, from Service Platform components to the network resources. The Traffic Shaping component Web service implements a distributed token bucket algorithm that allows for specifying the maximum burst and maximum sustained average rate of traffic emitted from the cluster.

### Network Resources component Web service

The Network Resources component Web service provides a way to define specifications for network resources (elements) with which the service platform will interact. Specifications that you design using the Network Resources component Web service are used to tailor the way in which the service platform behaves toward each resource.

### Notification Management component Web service

The Notification Management component Web service, captures information about active notifications on the service platform for use by administrators who are monitoring the network. It provides functionality that administers registration, termination, and query notifications.

The Notification Management component Web service uses the DB2 or Oracle databases, which must be initialized by using the First Steps script.

### PX Notification component Web service

This component is used by several Web service implementations as a support service for delivering outbound Web service notifications This component uses a JMS queue (which is optional), which requires a service integration bus called TWSSBus.

### Address Masking component Web service

The Address Masking component Web service can be used to hide the real identity of a subscriber from any third-party application (TPA).

The Address Masking component Web service has been deployed as a Service Platform component, and the service policy values required for Address Masking needs to be configured in the Service Policy Manager. The version 7.0 Access Gateway and Service Policy Manager both are required prerequisites for using this feature.

### Fault and Alarm component Web service

The Fault and Alarm component Web service can be deployed to capture faults and alarms that occur for each deployed Web service implementation, for the Admission Control component Web service, and for the Usage Record component Web service. It can write fault and alarm information to the common event infrastructure (CEI) repository. If you plan to use the CEI repository to capture fault and alarm data, you must enable the CEI service and configure a data source for it.

The CEI service uses the database included with WebSphere Application Server, but it can be configured to use DB2®, Oracle or another database supported by WebSphere Application Server.

### Usage Record component Web service

The Usage Record component Web service writes usage information to one of the supported databases, where it could be retrieved for use by billing applications.

# Planning to install Web service implementations

The Telecom Web Services Server (TWSS) product supports a large number of Web service implementations, many of which are based on Parlay X specifications. The Web services implementations that are needed for your network will need to be deployed. This section explains what needs to be installed in your network for a given Web service.

The information in the following topics is provided to help you understand the elements that must be present in the network environment for successful deployment of specific Web service implementations. To understand the order in which components should be installed, refer to the installation scenarios.

**Note:** Some of the Web service implementations have unique migration considerations when you are moving from TWSS version 6.2 to version 7.0. For details, refer to the topic *Planning to migrate from a previous release of Telecom Web Services Server*.

## Integrating the Web service implementations with other components

To make full use of the Web service implementations that are available with TWSS, deploy them to use the corresponding Web service APIs. Each Web service implementation can coexist with other network components. Considerations for setting up this coexistence vary, depending on which Web service implementations you plan to deploy.

### Integrating with Service Platform components

The Service Platform components facilitate rapid development of the Web service implementations. You can deploy the Service Platform components and then disable any that are unused.

**Note:** For more information on how to disable service platform components, see the topic *Disabling Service Platform components for Web services* in the Administering section of the information center.

The following table summarizes which Service Platform components should be deployed with each Web service implementation. Install the Service Platform components locally on each application server in a cluster where the Web service implementations are deployed.

Notes about the table:

1. You should deploy the Traffic Shaping component Web service and the Network Resources component Web service whenever you plan to use the Web

service implementations in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist.

2. You should deploy both the Notification Management component Web service and the PX Notification component Web service together with any Web service implementation that generates or handles notifications. (There is one exception: for WAP Push over SMPP, only PX Notification is required.)

*Table 2. Interoperability between Web service implementations and Service Platform components*

| **Web service** | **Admission Control** | **Traffic Shaping (see note 1)** | **Network Resources (see note 1)** | **Notification Management** | **PX Notification** | **Address Masking** | **Fault and Alarm** | **Usage Record** |
|---|---|---|---|---|---|---|---|---|
| Call Notification (SIP) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Optional |
| Presence | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Recommended |
| Terminal Status (SIP) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Recommended |
| Call Handling (SIP/IMS) | Recommended | Optional | Recommended | Optional | Optional | Optional | Recommended | Recommended |
| Third Party Call (SIP) | Optional | Optional | Optional | Optional | Optional | Optional | Recommended | Recommended |
| Payment | Optional | Optional | Optional | Optional | Optional | Optional | Recommended | Required |
| Address List | Recommended | Optional | Optional | Optional | Optional | Optional | Recommended | Recommended |
| WAP Push | Optional | Recommended | Recommended | Optional | Required (see note 2) | Optional | Required | Recommended |
| SMS (SMPP) | Optional | Recommended | Recommended | Required (see note 2) | Required (see note 2) | Optional | Required | Recommended |
| Terminal Location (MLP) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Optional |
| MMS (MM7) | Optional | Optional | Optional | Required | Required | Optional | Required | Optional |
| Terminal Status (Parlay) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Optional |
| Terminal Location (Parlay) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Optional |
| SMS (Parlay) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Required | Optional |

*Table 2. Interoperability between Web service implementations and Service Platform components (continued)*

| **Web service** | **Admission Control** | **Traffic Shaping (see note 1)** | **Network Resources (see note 1)** | **Notification Management** | **PX Notification** | **Address Masking** | **Fault and Alarm** | **Usage Record** |
|---|---|---|---|---|---|---|---|---|
| Call Handling (Parlay) | Recommended | Optional | Recommended | Optional | Optional | Optional | Recommended | Optional |
| Call Notification (Parlay) | Optional | Optional | Optional | Required (see note 2) | Required (see note 2) | Optional | Recommended | Optional |
| Third Party Call (Parlay) | Optional | Optional | Optional | Optional | Optional | Optional | Recommended | Recommended |

## Integrating with system administration consoles

The following is required for integrating the system administration console.

- The TWSS Administration Console is required if you plan to deploy any of the following:
  - Web service implementations based on SIP/IMS
  - The Parlay X Payment (PostPaid) Web service implementation
  - The Parlay X Address List Manager over XCAP Web service implementation
  - Web service implementations based on Direct Connect protocols
  - The Service Platform components

Use the TWSS Administration Console to manage and configure the Service Platform components and Web service implementations.

Use the Parlay Administration Console to configure the Parlay Connector. The Parlay Connector is needed only for the Parlay-based Web service implementations:

- Parlay X Terminal Status over Parlay
- Parlay X Terminal Location over Parlay
- Parlay X SMS over Parlay
- Parlay X Call Handling over Parlay
- Parlay X Call Notification over Parlay
- Parlay X Third Party Call over Parlay

**Note:** Both the TWSS and Parlay consoles are never installed on the node, instead they are installed on the dmgr machine. The administration consoles are distinct from, and should not be confused with, the Service Policy Manager console.

## Integrating with a privacy Web service

Each Web service implementation has the ability to integrate with a privacy service. When each Web service implementation is deployed, the privacy component endpoint is enabled in order to protect the Web service implementation and prevent unauthorized access. If a privacy service does not exist in the network, disable the privacy client. For more information, refer to the topic *Configuring privacy integration*.

### Performance

The performance of any given Web service implementation is influenced by several factors including the performance of the network, the performance of the database, and on the way in which the front end to the network (which, in the case of TWSS, is the Access Gateway) is configured.

## Planning checklist for Web service implementations

Before you begin the installation, decide where you will install the WebSphere Telecom Web Services Server components. Print this page and record information about your environment before you begin.

### General coexistence considerations

The following Direct Connect services can coexist in any combination, for instance:

- Parlay X SMS over SMPP
- WAP Push over SMPP
- Parlay X Terminal Location over MLP
- Parlay X Multimedia Messaging over MM7

Multiple service implementations for any of the following Web services cannot be deployed together. For example Parlay X Call Notification over SIP/IMS cannot be deployed together with Parlay X Call Notification over Parlay.

- Call Notification
- Call Handling
- Terminal Location
- Terminal Status
- Third Party Call
- Short Message Service (SMS)

### Database tables

Refer to the topic *Database tables for Web service implementations* for a list of the database table names that are required for each Web service implementation.

### Parlay X Call Notification over SIP/IMS

If you plan to deploy Parlay X Call Notification over SIP/IMS, you will need the following in your network.

- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- An S-CSCF or SIP proxy to direct call establishment (optional).
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Database tables to store

  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - CallNotificationAddress table
- A JMS queue (which requires a service integration bus, TWSSBus) for the PX Notification component Web service.
- A shared file system space to store the JMS queue file store.

## Parlay X Presence over SIP/IMS

If you plan to deploy Parlay X Presence over SIP/IMS, you will need the following in your network.

- IBM WebSphere Presence Server Component or other IMS-compliant presence server
- IBM WebSphere XML Document Management Server Component to resolve groups
- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- Billing mediator to process usage records or CEI events
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- A JMS queue (which requires a service integration bus, TWSSBus) for the PX Notification component Web service.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service, if you plan to use it
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

## Parlay X Terminal Status over SIP/IMS

If you plan to deploy Parlay X Terminal Status over SIP/IMS, you will need the following in your network.

- An S-CSCF or SIP proxy to direct call establishment (optional).
- IBM WebSphere Presence Server Component or other IMS-compliant presence server
- IBM WebSphere XML Document Management Server Component to resolve groups

- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- A JMS queue (which requires a service integration bus, TWSSBus) for the PX Notification component Web service.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service, if you plan to use it
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

## Parlay X Third Party Call over SIP/IMS

If you plan to deploy Parlay X Third Party Call over SIP/IMS, you will need the following in your network.

- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- An S-CSCF or SIP proxy to direct call establishment (optional).
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Data for the third party call information
- Optional: Consider the endpoints for UserAgent PBX numbers or SIP Software IP addresses.

## Parlay X Payment (PostPaid)

If you plan to deploy Parlay X Payment (PostPaid), you will need the following in your network.

- Billing mediator to process usage records or CEI events
- Access Gateway

- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

## Parlay X Address List Manager over XCAP

If you plan to deploy Parlay X Address List Manager over XCAP, you will need the following in your network.

- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- IBM WebSphere XML Document Management Server Component (IBM XDMS), or equivalent XCAP server, deployed and configured in the TWSS Administration Console
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

The Web service implementation needs to be aware of the security settings for the XCAP server: either the Trust Association Interceptor, basic authentication, or unauthenticated.

## WAP Push over SMPP

If you plan to deploy WAP Push over SMPP, you will need the following in your network:

- Short Message Service Center (SMSC)
- SMPP connector
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console

- JMS queues (which require a service integration bus, TWSSBus) for the Web service
- A shared file system space to store the JMS queue file store.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - WAP Push data
  - TWSS Administration Console configuration data
  - Notification data
  - Data for the Usage Record component Web service, if you plan to use it
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

## Parlay X SMS over SMPP

If you plan to deploy Parlay X SMS over SMPP, you will need the following in your network.

- Short Message Service Center (SMSC)
- SMPP connector
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- JMS queues (which require a service integration bus, TWSSBus) for the PX Notification component Web service and for the Web service itself

  **Note:** This is not required for synchronous messaging.
- A shared file system space to store the JMS queue file store.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - SMS data

## Parlay X Terminal Location over MLP

If you plan to deploy Parlay X Terminal Location over MLP, you will need the following in your network.

- Access Gateway
- Service Policy Manager

- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- A JMS queue (which requires a service integration bus, TWSSBus) for the PX Notification component Web service.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Terminal location data

## Parlay X Multimedia Messaging over MM7

If you plan to deploy Parlay X Multimedia Messaging over MM7, you will need the following in your network.

- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- JMS queues (which require a service integration bus, TWSSBus) for the PX Notification component Web service and for the Web service itself
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - MMS data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

## Parlay X Terminal Status over Parlay

If you plan to deploy Parlay X Terminal Status over Parlay, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console

- A JMS queue (which requires a service integration bus, TWSSBus) for the PX Notification component Web service.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service, if you plan to use it
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive

## Parlay X Terminal Location over Parlay

If you plan to deploy Parlay X Terminal Location over Parlay you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console
- A JMS queue (which requires a service integration bus, TWSSBus) for the PX Notification component Web service.
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service, if you plan to use it
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Notification data

## Parlay X SMS over Parlay

If you plan to deploy Parlay X SMS over Parlay, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- JMS queues (which require a service integration bus, TWSSBus) for the PX Notification component Web service and for the Web service itself
- Parlay Administration Console

- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - SMS data

## Parlay X Call Handling over Parlay

If you plan to deploy Parlay X Call Handling over Parlay, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console
- JMS queue (which requires a service integration bus, TWSSBus), for the PX Notification component Web service.
- Shared file system space to store the JMS queue file store
- Database tables to store:
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Data for Call Handling rules

## Parlay X Call Handling over Parlay

If you plan to deploy Parlay X Call Handling over Parlay, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console
- JMS queue (which requires a service integration bus, TWSSBus), for the PX Notification component Web service.

- Shared file system space to store the JMS queue file store
- Database tables to store:
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Data for Call Handling rules

## Parlay X Call Handling over SIP/IMS

If you plan to deploy Parlay X Call Handling over SIP/IMS, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console
- JMS queue (which requires a service integration bus, TWSSBus), for the PX Notification component Web service.
- Shared file system space to store the JMS queue file store
- Database tables to store:
  - Transaction and network statistics data
  - Policy data
  - Notification data
  - TWSS Administration Console configuration data
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Data for Call Handling rules

## Parlay X Call Notification over Parlay

If you plan to deploy the Parlay X Call Notification over Parlay, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console
- Database tables to store
  - Transaction and network statistics data

  - Policy data
  - TWSS Administration Console configuration data
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - CallNotification data

### Parlay X Third Party Call over Parlay

If you plan to deploy the Parlay X Third Party Call over Parlay, you will need the following in your network.

- Parlay Connector, to communicate with a Parlay gateway
- Access Gateway
- Service Policy Manager
- Service Platform components (see the topic *Required components for the Web service implementations*).
- TWSS Administration Console
- Parlay Administration Console
- Database tables to store
  - Transaction and network statistics data
  - Policy data
  - TWSS Administration Console configuration data.
  - Data for the Usage Record component Web service
  - Traffic data for use by the SLA Enforcement mediation primitive
  - Information about messages, used by the Message Interceptor mediation primitive
  - Call status information

## Planning for Web service implementations based on SIP/IMS

Telecom Web Services Server supports Web service implementations that operate over SIP/IMS.

### Planning to install Parlay X Call Notification over SIP/IMS

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Call Notification over SIP/IMS.

This topic provides information about what is needed in the network in order to deploy the Web service. The order in which the information is presented does not represent the order in which the elements should be installed.

Your network must include the following elements:

- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- An S-CSCF or SIP proxy to direct call establishment (optional).
- A shared file system space to store the JMS queue file store.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Call Notification message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 3. Service Platform components for Parlay X Call Notification over SIP/IMS*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | The `startCallNotification` operation issues a `RegisterNotification` request. The `stopCallNotification` and `terminateNotification` operations issue a `RemoveNotification` request. |

*Table 3. Service Platform components for Parlay X Call Notification over SIP/IMS (continued)*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | Unlike the other Service Platform components, this Web service is utilized only by the Address Masking mediation primitive of the Access Gateway. It is used to mask the subscriber address in a message before the message is sent to a third-party application. In addition, messages from third-party applications are processed so that any masked addresses they contain are unmasked before the messages are sent downstream. |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Call Notification over SIP/IMS fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs<br>• A Notification Management component Web service fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## JMS queues

A JMS queue is required for the PX Notification component Web service, however there are also several other service implementations which use JMS queues.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Call Notification over SIP/IMS Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- SIP proxy: A SIP proxy must be created and configured for requests to the IMS control plane. WebSphere Application Server includes a SIP proxy server that can be used.
- S-CSCF: Parlay X Call Notification over SIP/IMS uses the S-CSCF network element as a SIP proxy for outbound TEL and SIP URIs.

## Planning to install Parlay X Presence over SIP/IMS

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Presence over SIP/IMS.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

Your network must include the following elements:

- IBM WebSphere Presence Server Component or other IMS-compliant presence server.
- IBM WebSphere XML Document Management Server Component to resolve groups.
- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- Billing mediator to process usage records or CEI events.
- SIP proxy

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Presence message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

The optional group resolution mediation primitive in the default flow is used when group information is included with the message. The group resolution mediation primitive is required and it resolves group URIs. If you are using a custom message processing flow that does not include the group resolution mediation primitive, incoming URIs are processed like a single presentity URI.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 4. Service Platform components for Parlay X Presence over SIP/IMS*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | Used by the `startPresenceNotification` and `endPresenceNotification` operations. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |

*Table 4. Service Platform components for Parlay X Presence over SIP/IMS (continued)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Presence over SIP/IMS fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs<br>• A Notification Management component Web service fault occurs.<br>• A database-related CEI repository fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## JMS queues

When installing Parlay X Presence over SIP/IMS, a JMS queue is required for the PX Notification component Web service.

When setting up the service integration buses, the wizard will configure a permanent file store. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you.

## Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Presence over SIP/IMS Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- SIP proxy: A SIP proxy must be created and configured for requests to the IMS control plane. WebSphere Application Server includes a SIP proxy server that can be used.
- IBM WebSphere Presence Server: The Parlay X Presence over SIP/IMS sends a SIP SUBSCRIBE request to the Presence Server to retrieves presence information. The Parlay X Presence over SIP/IMS processes and converts the corresponding SIP NOTIFY message into Parlay X data. The Parlay X Presence over SIP/IMS interacts with Presence Server through the Pw, Pet, Pep, and Peu reference points
- IBM WebSphere XML Document Management Server Component (IBM XDMS): Parlay X Presence over SIP/IMS can use IBM XDMS to resolve groups. To use IBM XDMS, the message processing flow must use the group resolution mediation primitive.
- Privacy service: If the privacy client is enabled and integrated with a privacy service, Parlay X Presence over SIP/IMS will use it to determine if a particular Parlay X client is authorized to access the presence information for the presentity. The Parlay X specification defines a reactive model, but Parlay X Presence over SIP/IMS can use a proactive model, if a privacy service is available.

## Planning to install Parlay X Terminal Status over SIP/IMS

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Terminal Status over SIP/IMS.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

When you deploy this Web service implementation, your network should include the following elements:

- The WebSphere Application Server platform
- An S-CSCF or SIP proxy to direct call establishment (optional).
- IBM WebSphere Presence Server Component or other IMS-compliant presence server.
- IBM WebSphere XML Document Management Server Component to resolve groups.
- TheWebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- A shared file system space to store the JMS queue file store

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Terminal Status message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 5. Service Platform components for Parlay X Terminal Status over SIP/IMS*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | Used by the `StartNotification` and `EndNotification` operations. |

*Table 5. Service Platform components for Parlay X Terminal Status over SIP/IMS (continued)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Terminal Status over SIP/IMS fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs<br>• A Notification Management component Web service fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## JMS queues

When installing Parlay X Terminal Status over SIP/IMS, a JMS queue is required for the PX Notification component Web service.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

## Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Terminal Status over SIP/IMS Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- SIP proxy: A SIP proxy must be created and configured for requests to the IMS control plane.WebSphere Application Server includes a SIP proxy server that can be used.
- S-CSCF: Parlay X Terminal Status over SIP/IMS uses the S-CSCF network element as a sip proxy for outbound tel and sip URIs.
- WebSphere Presence Server: The Parlay X Terminal Status over SIP/IMS retrieves information about the status if a terminal from the Presence Server. It uses the Pw reference point to interact with the Presence Server.
- IBM WebSphere XML Document Management Server Component (IBM XDMS): Parlay X Presence over SIP/IMS can use IBM XDMS to resolve groups. To use IBM XDMS, the message processing flow must use the group resolution mediation primitive.
- Privacy service: If there is a privacy service, Parlay X Terminal Status over SIP/IMS integrates with it to determine if a particular Parlay X client is authorized to access the status for a given terminal.

## Planning to install Parlay X Call Handling over SIP/IMS

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Call Handling over SIP/IMS.

When you deploy this Web service implementation, the only required element in your network is the WebSphere Application Server platform.

**Note:** The order in which the following sections are presented is not necessarily the order in which you will need to set up your system.

### Access Gateway

Before the Parlay X Call Handling over SIP/IMS Web service is called, the service authorization mediator in the Telecom Web Services Access Gateway flow, will authenticate your access.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

## TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 6. Service Platform components for Parlay X Call Handling over SIP/IMS*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |

*Table 6. Service Platform components for Parlay X Call Handling over SIP/IMS (continued)*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Call Handling over SIP/IMS fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway. The Parlay Connector is included in the file `Pxx_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

## Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

## Other interactions

The Parlay X Call Handling over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important.

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X Call Handling over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning to install Parlay X Third Party Call over SIP/IMS

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Third Party Call over SIP/IMS.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

Your network should include the following elements:
- The WebSphere Application Server proxy, to handle requests to the Web service and to the IMS control plane.
- An S-CSCF or SIP proxy to direct call establishment (optional).

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Third Party Call message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 7. Service Platform components for Parlay X Third Party Call over SIP/IMS*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | No | Not used. |
| PX Notification | No | Not used. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Third Party Call over SIP/IMS fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs<br>• A Notification Management component Web service fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes.<br>A usage record is generated for the makeCall, endCall, getCallInformation and cancelCallRequest event. The usage record includes information about how many calls were made, how long the calls lasted, and whether there were any associated charges. |

*Table 7. Service Platform components for Parlay X Third Party Call over SIP/IMS (continued)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Privacy | No | When deployed, a privacy Web service allows for integration with the Parlay X Third Party Call over SIP/IMS Web service. |

### Service integration

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Third Party Call over SIP/IMS Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- SIP proxy: A SIP proxy must be created and configured for requests to the IMS control plane. WebSphere Application Server includes a SIP proxy server that can be used.
- S-CSCF (optional): Parlay X Third Party Call over SIP/IMS uses the S-CSCF network element as a sip proxy for outbound TEL and sip URIs.

## Planning for Web service implementations based on IBM WebSphere software

Telecom Web Services Server (TWSS) supports Web service implementations that interoperate with other components within the IBM WebSphere product group–including WebSphere Application Server, IBM XDMS, and other instances of TWSS.

## Planning to install Parlay X Payment (PostPaid)

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Payment (PostPaid).

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

Your network must include the following element:

- Billing mediator to process usage records or CEI events

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Payment message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 8. Service Platform components for Parlay X Payment (PostPaid)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | No | Not used. |
| PX Notification | No | Not used. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Payment (PostPaid) fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A database-related CEI repository fault occurs. |
| Usage Record | Yes | Used to record information about services rendered for status and billing purposes. |

## Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Payment (PostPaid) Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Billing mediator: A billing mediator reads database records, information written by Parlay X Payment (PostPaid), and generates billing events to your billing system.

## Planning to install Parlay X Address List Manager over XCAP

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Address List Manager over XCAP.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

Parlay X Address List Manager over XCAP interacts by default with the IBM WebSphere XML Document Management Server Component (IBM XDMS) and can work with any XCAP server.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

For information about optional configuration of IBM XDMS and XCAP, refer to the IBM XDMS portion of this information center.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 9. Service Platform components for Parlay X Address List Manager over XCAP*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | Recommended. When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Not used. |
| Network Resources | No | Not used. |
| Notification Management | No | Not used. |
| PX Notification | No | Not used. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Address List Manager over XCAP fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Database

Parlay X Address List Manager over XCAP uses the database to store MBean related data. A script is provided to initialize the database table.

### Other interactions

The Parlay X Address List Manager over XCAP Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- XDMS: Parlay X Address List Manager over XCAP interacts with the IBM WebSphere XML Document Management Server Component (IBM XDMS). Consequently, the message processing flow must use the Group Resolution mediation primitive.

# Planning for Web service implementations based on Direct Connect protocols

Telecom Web Services Server supports Web service implementations that operate over Direct Connect protocols such as SMPP, MLP, and MM7.

## Planning to install WAP Push over SMPP

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy WAP Push over SMPP.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default WAP Push message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 10. Service Platform components for WAP Push over SMPP*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | No | Not used. |
| PX Notification | No | Not used. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | Yes | Issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A WAP Push over SMPP fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## JMS Queues

When you deploy WAP Push over SMPP, multiple JMS queues are required for use by the Web service.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store WAP Push message data, configuration data, and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The WAP Push over SMPP Web service implementation accepts requests for sending WAP Push messages and controls various checks that are done to ensure the requests are valid. The process then forwards the requests to the appropriate network elements for downstream processing. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- SMSC: WAP Push over SMPP connects with the SMS message center and sends WAP Push messages to it.
- SMPP JCA Adapter: SMPP connector passes every message to the SMSC as it arrives to the SMPP connector layer.

## Planning to install Parlay X SMS over SMPP

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X SMS over SMPP.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

**Note:** Do not deploy Parlay X SMS over SMPP on the same application server as Parlay X SMS over Parlay.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default SMS message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 11. Service Platform components for Parlay X SMS over SMPP*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | The `startNotification` operation issues a `RegisterNotification` request. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |

*Table 11. Service Platform components for Parlay X SMS over SMPP (continued)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Fault and Alarm | Yes | Issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## JMS Queues

When you deploy Parlay X SMS over SMPP, multiple JMS queues are required. One is required for the PX Notification component Web service and the others are for Parlay X SMS over SMPP.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

## Database

Database tables must be created to store SMS data, configuration data, and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

## Other interactions

The Parlay X SMS over SMPP Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.

- SMSC: Parlay X SMS over SMPP interacts with the SMS message center. It sends data to and receives data from the SMSC. The data includes SMS logos and ring tones as well as messages.
- SMPP JCA Adapter: SMPP connector passes every message to the SMSC as it arrives to the SMPP connector layer.

## Planning to install Parlay X Terminal Location over MLP

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Terminal Location over MLP.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

Your network should include the following element:
- MLP location server

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Terminal Location message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 12. Service Platform components for Parlay X Terminal Location over MLP*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | Used by the `StartGeographicalNotification` and `StartPeriodicNotification` operations. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Terminal Location over MLP fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## JMS queues

When installing Parlay X Terminal Location over MLP, a JMS queue is required for the PX Notification component Web service.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Terminal Location over MLP Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.

## Planning to install Parlay X Multimedia Messaging over MM7

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Multimedia Messaging over MM7.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default MMS message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console

The console must be installed on the deployment manager node for the cluster where the Web service is deployed.The console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR files for the Web service implementations. It is managed through JMX.

The console plug-in acts as an extension to the WebSphere Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 13. Service Platform components for Parlay X Multimedia Messaging over MM7*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | Used by the `startMessageNotification` and `stopMessageNotification` operations. |
| PX Notification | Yes | The application is notified by means of a call to the Service Platform's PX Notification Delivery Web service notifyMessageDeliveryReceipt method. |
| Address Masking | No | This Web service is used only by the Access Gateway component's Address Masking mediation primitive. It is used to mask a subscriber address in a message before it is sent to a third-party application and to unmask addresses in messages received from third-party applications before sending them further downstream. |
| Fault and Alarm | Yes | Issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |
| Privacy | No | A privacy Web service is optional for handling privacy checks. |

### JMS queues

When you deploy Parlay X Multimedia Messaging over MM7, multiple JMS queues are required. One is required for the PX Notification component Web service and the others are for Parlay X Multimedia Messaging over MM7.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store MMS data, configuration data, and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Multimedia Messaging over MM7 Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Multimedia Message Service Center (MMSC): Parlay X Multimedia Messaging over MM7 sends messages to an MMSC using the MM7 interface for onward transmission.

# Planning for Parlay-based Web service implementations

Telecom Web Services Server supports Web service implementations that operate over Parlay 3.x and 4.x APIs with Parlay 5.0 Multimedia Messaging (MMM).

## Planning to install Parlay X Terminal Status over Parlay

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Terminal Status over Parlay.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

## Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Terminal Status message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

## Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

## TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 14. Service Platform components for Parlay X Terminal Status over Parlay*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |

*Table 14. Service Platform components for Parlay X Terminal Status over Parlay (continued)*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| Notification Management | Yes | Provides notification tracking and administrative functionality. Registered notifications are stored within a backing database and are made available for query or, optionally, termination through an administrative interface. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Terminal Status over Parlay fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway. The Parlay Connector is included in the file `Pxx_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

### JMS queues

When you deploy Parlay X Terminal Status over Parlay, a JMS queue is required for the PX Notification component Web service.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

Parlay X Terminal Status over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X Terminal Status over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning to install Parlay X Terminal Location over Parlay

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Terminal Location over Parlay.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

When you deploy this Web service implementation, your network should include the following elements:

- The WebSphere Application Server platform
- A shared file system space to store the JMS queue file store

## Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Terminal Location message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

## Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

## TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 15. Service Platform components for Parlay X Terminal Location over Parlay*

| **Service Platform component** | **Required?** | **Remarks** |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Not used. |
| Network Resources | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |

Table 15. Service Platform components for Parlay X Terminal Location over Parlay (continued)

| Service Platform component | Required? | Remarks |
|---|---|---|
| Notification Management | Yes | The startNotification operation issues a RegisterNotification request. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Terminal Location over Parlay fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway. The Parlay Connector is included in the file `Pxx_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

## JMS queues

When you deploy Parlay X Terminal Location over Parlay, a JMS queue is required for the PX Notification component Web service.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Terminal Location over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X Terminal Location over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning to install Parlay X SMS over Parlay

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X SMS over Parlay.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

When you deploy this Web service implementation, your network should include the following elements:

- The WebSphere Application Server platform
- A shared file system space to store the JMS queue file store

Do not deploy Parlay X SMS over Parlay on the same application server as Parlay X SMS over SMPP.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default SMS message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 16. Service Platform components for Parlay X SMS over Parlay*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |

Table 16. Service Platform components for Parlay X SMS over Parlay (continued)

| Service Platform component | Required? | Remarks |
|---|---|---|
| Notification Management | Yes | The startNotification operation issues a RegisterNotification request. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | Yes | Issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X SMS over Parlay fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs. |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway. The Parlay Connector is included in the file `P`*`xx`*`_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

## JMS queues

When installing the Parlay X SMS over Parlay, you must configure multiple JMS queues. One is required for the PX Notification component Web service and the other is for the Parlay X SMS over Parlay.

**Note:** For example; *PX21_SMS_ParlayQueue* and *SP_PX21_PXNOTIFYQueue*.

When setting up the service integration buses, the First Steps script will configure a permanent file store or datastore. If you choose to configure the file store manually, the file store should be stored on a mounted NFS drive. Each instance of the Web service in the cluster should use the same permanent store directory path.

**Note:** When you use the First Steps script to automate your configuration, the service integration buses are defined for you. However, you will need to provide a file system directory on the NFS shared drive.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X SMS over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important:
- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X SMS over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning to install Parlay X Call Handling over Parlay

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Call Handling over Parlay.

When you deploy this Web service implementation, the only required element in your network is the WebSphere Application Server platform.

**Note:** The order in which the following sections are presented is not necessarily the order in which you will need to set up your system.

### Access Gateway

Before the Parlay X Call Handling over Parlay Web service is called, the service authorization mediator in the Telecom Web Services Access Gateway flow, will authenticate your access.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

## TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 17. Service Platform components for Parlay X Call Handling over Parlay*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |

*Table 17. Service Platform components for Parlay X Call Handling over Parlay (continued)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Call Handling over Parlay fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A PX Notification component Web service fault occurs |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway. The Parlay Connector is included in the file `Pxx_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

## Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

## Other interactions

The Parlay X Call Handling over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important.

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X Call Handling over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning to install Parlay X Call Notification over Parlay

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Call Notification over Parlay.

When you deploy this Web service implementation, your network should include the following elements:

- The WebSphere Application Server platform
- A shared file system space to store the JMS queue file store

**Note:** The order in which the following sections are presented is not necessarily the order in which you will need to set up your system.

### Access Gateway

TheAccess Gateway must be installed in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Call Notification message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform components cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

## Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 18. Service Platform components for Parlay X Call Notification over Parlay*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | Yes | Registers and removes notifications. |
| PX Notification | Yes | The Web service implementation sends notification messages asynchronously to applications through this component. A retry mechanism has been added to the notification delivery of Parlay X services. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |
| Fault and Alarm | No | Recommended. |
| Usage Record | No | When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes. |

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway. The Parlay Connector is included in the file `Pxx_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

### Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Call Notification over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important:

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X Call Notification over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning to install Parlay X Third Party Call over Parlay

Here is a summary of the TWSS components, database tables, and other network elements that are required when you deploy Parlay X Third Party Call over Parlay.

This topic provides information about what is needed in the network in order to deploy the Web service implementation. The order in which the information is presented does not represent the order in which the elements should be installed.

Your network should include the following elements:

- The WebSphere Application Server proxy, to handle requests to the Web service
- The Parlay Connector to communicate with a Parlay gateway

Do not deploy Parlay X Third Party Call over Parlay on the same application server as Parlay X Third Party Call over SIP/IMS.

### Access Gateway

The Access Gateway must be installed in the network and is typically in a different cluster than the Web service implementation. All incoming calls are routed through message processing flows, which are deployed on the Access Gateway. A default Third Party Call message processing flow is provided and should be deployed, unless you are using a custom message processing flow.

### Service Policy Manager

The Service Policy Manager must be installed in the network, either in the Service Platform cluster or in its own cluster. The Service Policy Manager manages policies associated with the Web service implementation. When the database tables are created, they are populated with default policies. For information about default policies, see the Reference section of this information center.

### TWSS Administration Console and Parlay Administration Console

The consoles must be installed on the deployment manager node for the cluster where the Web service is deployed. The TWSS Administration Console must be installed before the Parlay Administration Console.

The TWSS Administration Console is used to configure the Web service implementations and the Service Platform components and to specify how they should be invoked. It is also used to administer MBeans that are included in the EAR file for the Web service implementations. It is managed through JMX.

The Parlay Administration Console is used to configure the Parlay Connector.

Both console plug-ins act as extensions to the Integrated Solutions Console.

### Service Platform components

The Service Platform components must be installed on the same cluster node as the Web service implementation. If the cluster is configured to synchronize, you can deploy the Service Platform components on the network deployment manager node. Some Service Platform components may be optional and, if so, are noted as optional in the following list. Interactions with other Service Platform components are also noted.

*Table 19. Service Platform components for Parlay X Third Party Call over Parlay*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Admission Control | No | When the Admission Control component Web service is available, the Web service implementation issues a `VerifyAdmittableRequest` to the Admission Control component to determine if the service request can be processed. |
| Traffic Shaping | No | Recommended in an integrated environment–that is, in a full setup where all of the TWSS components (Access Gateway, Service Policy Manager, and so forth) coexist. |
| Network Resources | No | Recommended in an integrated environment. |
| Notification Management | No | Not used by Parlay X Third Party Call over Parlay. |
| PX Notification | No | Not used by Parlay X Third Party Call over Parlay. |
| Address Masking | No | When deployed, the Address Masking component Web service works in conjunction with the Address Masking mediation primitive (part of the processing flow for the Access Gateway). |

*Table 19. Service Platform components for Parlay X Third Party Call over Parlay (continued)*

| Service Platform component | Required? | Remarks |
|---|---|---|
| Fault and Alarm | No | Recommended. When the Fault and Alarm component Web service is available, the Web service implementation issues a `RecordFaultRequest` request to the Fault and Alarm component to generate a fault or an alarm when the following conditions occur:<br>• A Parlay X Third Party Call over Parlay fault condition is detected.<br>• An Admission Control component Web service fault occurs.<br>• A Usage Record component Web service fault occurs.<br>• A database-related CEI repository fault occurs. |
| Usage Record | No | Recommended. When the Usage Record component Web service is available, the Web service implementation uses it to record information about services rendered for status and billing purposes.<br>A usage record is generated for the `makeCall` event and for the `endCall` event. The usage record includes information about how many calls were made, how long the calls lasted, and whether there were any associated charges. |

Parlay X Third Party Call over Parlay uses the Service Platform's Integrated Solutions Console interface and MBeans to provide configuration settings. The wsadmin for MBeans is also available for the configuration settings.

## Parlay Connector

The Parlay Connector enables the Web services to communicate with a Parlay gateway. In a production environment, an additional Parlay gateway may be configured for failover. When you have multiple Parlay gateways in your network, you must configure unique service IDs for each gateway. The Web service implementations use the gateway session IDs to track Parlay X Third Party Call over Parlay requests sent to the gateway. The Parlay Connector is included in the file `Pxx_ParlayConnector.ear` (where *xx* is the Parlay version, such as 42 or 33).

To configure the Parlay Connector, use the Parlay Administration Console. The Parlay Administration Console must be installed on the network deployment manager for the cluster where the Web service is deployed.

## Database

Database tables must be created to store configuration and policy data. You can use a single database to store this data, or you can create a unique database for each type of data. When you use the First Steps script, the required tables are created for a single database configuration.

If you choose to use multiple unique databases, or if you prefer to configure the database tables manually, you can find additional information in the topic *Planning for the database*.

DB2 and Oracle databases are supported.

### Other interactions

The Parlay X Third Party Call over Parlay Web service implementation interacts with network elements and other entities. The following interactions are important.

- HTTP proxy: An HTTP proxy needs to be configured. WebSphere Application Server includes an HTTP proxy that can be used for this purpose.
- Parlay gateway: The Parlay gateway is a server that hosts the service implementations for the Parlay API. Parlay X Third Party Call over Parlay connects to it by means of the Parlay Connector. The Parlay API consists of various telecom service APIs, which provide an abstract interface to network elements deployed in your network. The Parlay Connector converts Java, the language of the Web service implementation, to CORBA, the language of the gateway.

## Planning for the database

The various subcomponents of theTelecom Web Services Server consists of Access Gateway, Service Policy Manager, Service Platform components, and the TWSS Administration Console. These store information about transactions, services, and configurations in one or more databases. You have several options for choosing the database configuration that is best suited for you.

**Note:** The DDLs are modified so that the DROP statements are commented out generally. This occurs so that the firststeps process has the ability to drop the tables selectively, depending on the configuration mode. If you manually use a DDL file to create a distributed database, you can uncomment the DROP statements in the file if you desire to recreate the database tables.

Telecom Web Services Server supports connections to both Oracle and DB2 databases.

The installation instructions in this information center assume that you are using one of two database configurations:

- *Consolidated* or *shared*, in which all of the tables (including the Parlay Connector tables) are created into a single database instance. This is the default configuration to get a single Telecom Web Services Server system up and running.
- *Distributed*, achieved by first initializing the system using a consolidated database and then migrating particular tables to other database instances as desired.

When setting up the database server, you should include failover mechanisms. Also, use a high performance server with sufficient Disk I/O capabilities for your database server.

## Database tables for Telecom Web Services Server subcomponents

The base subcomponents of Telecom Web Services Server require several different database tables. You can configure your system so that the database tables reside in a single, shared database or in multiple databases. After you create your database or databases, the tables are defined when you perform initial configuration using the First Steps script.

Most Telecom Web Services Server installations use a *shared* or *consolidated* database configuration. In this configuration, database tables for all Telecom Web Services Server subcomponents are created in one database, typically called TWSSDB. Additionally, the tables for the Parlay Connector is now also a part of the same database. Multiple data sources are now provided for ease of modifying to a distributed topology.

Alternatively, you can use a *distributed* database configuration in which database tables for all Telecom Web Services Server subcomponents are created in multiple database instances. Using this configuration can improve performance and prevent bottlenecks when accessing the database tables.

During the standard installation and configuration process, the First Steps script sets up a shared database configuration by default.

The following tables list the names of the database instances and other information for TWSS subcomponents using both the shared configuration and the distributed configuration.

### Shared configuration

In a shared or consolidated configuration, database tables for the various subcomponents reside in a single database commonly called TWSSDB. This database also can include tables for the Parlay Connector, if you plan to deploy any of the Parlay-based Web service implementations.

For each component, the following table lists:

- Name of the associated database instance.
- Database Definition Language (DDL) files used by the First Steps script or by the database setup scripts (`crtsrv`*`xx`*) to create or update the database tables.

  **Note:** Each subcomponent has separate DDL files for DB2 and Oracle.
- Required database tables.

When you are configuring for a shared database, the `BaseDb`*`xxx`*`.ddl` file creates database tables for all of the base components: Access Gateway, Service Policy Manager, TWSS Administration Console, and Service Platform components.

Refer to the topic *Database tables for Web service implementations* for a list of the database tables that are required for the Web service implementations.

*Table 20. Database tables for TWSS components (shared database configuration)*

| Component | Sample DB instance | DDLs | Database tables created | Notes® |
|---|---|---|---|---|
| Access Gateway | TWSSDB | BaseDbDb2.ddl<br>BaseDbOra.ddl | TRANSACTIONS<br>NETWORKSTATISTICS<br>ATTACHMENT_CONTENTS<br>MESSAGELOG<br>TARGETLOG<br>SLADATA | For Oracle databases, the following views are created: MSGLOG and NETWORKSTATISTICSVIEW. |
| Service Policy Manager | TWSSDB | BaseDbDb2.ddl<br>BaseDbOra.ddl | REQUESTERS<br>SERVICES<br>OPERATIONS<br>POLICYTYPES<br>POLICIES | Stores policy data for the Web services. When you deploy each service, an associated initialization (DDL) populates the database tables. |
| TWSS Administration Console | TWSSDB | BaseDbDb2.ddl<br>BaseDbOra.ddl | CONFIGPROPERTIES | Network Resources component Web service also uses this database. |
| Usage Record component Web service | TWSSDB | BaseDbDb2.ddl<br>BaseDbOra.ddl | USAGERECORDS | |
| Notification Management component Web service | TWSSDB | BaseDbDb2.ddl<br>BaseDbOra.ddl | NOTIFICATIONS<br>CRITERIA<br>TARGET | Needed only when you deploy the Notification Management component Web service. |
| Address Masking component Web service | TWSSDB | AddressMaskingDb2.ddl<br>AddressMaskingDbOra.ddl | TRANSACTIONIDIDENTIFIER<br>INPUTVALUE<br>OUTPUTVALUE<br>REQUESTER<br>INTERNALEXPIRYEXTERNALEXPIRYEXECUTECOUNT | |
| Fault and Alarm component Web service | TWSSDB | None. | None. | Can be configured to use CEI, which uses a database. |
| Parlay Connector | TWSSDB | ParlayConnectorDb2.ddl<br>ParlayConnectorOra.ddl | PUBLISHEDDATA<br>CFGPROPERTIES<br>KEYGENERATOR<br>SERVERPROPERTIES<br>EVENTRESOURCE | Needed only when you deploy Parlay-based Web services<br><br>Stores configuration data for the Parlay Connector and the Parlay Administration Console. |

## Distributed configuration

In a distributed configuration, database tables for the various subcomponents reside in multiple databases.

The First Steps script creates tables for a single, shared database that is shared by all of the TWSS components. If you prefer to use a distributed database configuration, you will need to configure the databases manually as described in the topic *Migrating data in a distributed database configuration*.

For each component, the following table lists:

- Name of the associated database instance.
- Database Definition Language (DDL) files used by the First Steps script or by the database setup scripts (`crtsrv`*xx*) to create or update the database tables.

  **Note:** Each subcomponent has separate DDL files for DB2 and Oracle.
- Required database tables.

*Table 21. Database tables for TWSS components (distributed database configuration)*

| Component | Sample DB instance | DDLs | Database tables created | Notes |
|---|---|---|---|---|
| Access Gateway | AGDB,<br>ATTACHDB | ESBDbDb2.ddl<br>ESBDbOra.ddl | TRANSACTIONS<br>NETWORKSTATISTICS<br>ATTACHMENT_CONTENTS<br>MESSAGELOG<br>TARGETLOG<br>SLADATA | When creating a unique database during the installation, you must define a corresponding data source with a unique JNDI name.<br>For Oracle databases, the following views are created: MSGLOG and NETWORKSTATISTICSVIEW. |
| Service Policy Manager | SPMDB | SPMDbDb2.ddl<br>SPMDbOra.ddl<br>**Note:** If you are moving the Service Policy Manager database tables after an initial installation, the tables contain policy data that will also need to be moved. | REQUESTERS<br>SERVICES<br>OPERATIONS<br>POLICYTYPES<br>POLICIES | Stores policy data for the Web services. When you deploy each service, an associated initialization (DDL) populates the database tables.<br>When creating a unique database during the installation, you must define a corresponding data source with a unique JNDI name. |

*Table 21. Database tables for TWSS components (distributed database configuration) (continued)*

| Component | Sample DB instance | DDLs | Database tables created | Notes |
|---|---|---|---|---|
| TWSS Administration Console | ADMINDB | AdminDbDb2.ddl<br>AdminDbOra.ddl<br>**Note:** If you are moving the database table after using the First Steps script to perform initial configuration, the table will contain configuration data that also needs to be moved. | CONFIGPROPERTIES | When creating a unique database during the installation, you must define a corresponding data source with a unique JNDI name.<br>Network Resources component Web service also uses this database. |
| Usage Record component Web service | USAGEDB | UsageDbDb2.ddl<br>UsageDbOra.ddl | USAGERECORDS | You can either create a unique database for the usage record data or create the tables in a shared instance.<br>When creating a unique database during the installation, you must define a corresponding data source with a unique JNDI name. |
| Address Masking component Web service | AMDB | AddressMaskingDbDb2.ddl<br>AddressMaskingDbOra.ddl | | |
| Fault and Alarm component Web service | | Not applicable | Not applicable | Can be configured to use CEI, which uses a database. |
| Notification Management component Web service | NMDB | ServicePlatformDbDb2.ddl<br>notifymgmtDbDb2.ddl<br>ServicePlatformDbOra.ddl<br>notifymgmtDbOra.ddl | NOTIFICATIONS<br>CRITERIA<br>TARGET | Needed only when you deploy the Notification Management component Web service. |

*Table 21. Database tables for TWSS components (distributed database configuration) (continued)*

| Component | Sample DB instance | DDLs | Database tables created | Notes |
|---|---|---|---|---|
| Parlay Connector | PARLAYDB | ParlayConnectorDb2.ddl<br>ParlayConnectorOracle.ddl | PUBLISHEDDATA<br>CONFIGPROPERTIES<br>KEYGENERATOR<br>SERVERPROPERTIES<br>EVENTRESOURCE | Needed only when you deploy Parlay-based Web services<br><br>Stores configuration data for the Parlay Connector and the Parlay Administration Console. |

## Database tables and stored procedures for Web service implementations

The Web service implementations provided with Telecom Web Services Server require several different database tables and stored procedures. These entities are defined during the installation and configuration process.

During the standard installation and configuration process, the First Steps script sets up database tables and stored procedures as shown here. For each Web service implementation, the following are listed:

- Database Definition Language (DDL) files used by the First Steps script or by the database setup scripts (`crtsvr`*`xxx`*) to create or update the database tables and stored procedures.

  **Note:** The *xxx* can represent either Db2 (for DB2) or Ora (for Oracle).

  Each subcomponent has separate DDL files for DB2 and Oracle.
- Required database tables and stored procedures that are set up by the First Steps script.

The DDL files named `Admin`*`xxx`* are invoked automatically by the First Steps script. Should you ever need to run one of these files manually in a distributed database configuration, run it against the database instance that contains the CONFIGPROPERTIES table (which is used by the TWSS Administration Console)–not against the database instance that contains the tables used by the Web service implementation.

Refer to the topic *Database tables for Telecom Web Services Server subcomponents* for a list of the database tables that are required for the base components.

**Note:** You need to create tables and stored procedures only for the Web service implementations that you plan to deploy.

*Table 22. Database tables for Web services*

| **Web service** | **DDLs** | **Database tables created or updated** |
|---|---|---|
| Parlay X Call Notification over SIP/IMS | CallNotifyDbDb2.ddl<br>AdminCnDb2.ddl<br>CallNotifyDbOra.ddl<br>AdminCnOra.ddl | CALLNOTIFICATIONADDRESS |
| Parlay X Presence over SIP/IMS | AdminPresDb2.ddl<br>AdminCmnDbDb2.ddl<br>AdminPresOra.ddl<br>AdminCmnDbOra.ddl | None |
| Parlay X Terminal Status over SIP/IMS | AdminTsDb2.ddl<br>AdminCmnDbDb2.ddl<br>AdminTsOra.ddl<br>AdminCmnDbOra.ddl | None |
| Parlay X Third Party Call over SIP/IMS | ThirdPartyDbDb2.ddl<br>AdminTpcDb2.ddl<br>ThirdPartyDbOra.ddl<br>AdminTpcOra.ddl | THIRDPARTYCALL |
| Parlay X Payment (PostPaid) | AdminPmtDb2.ddl<br>AdminPmtOra.ddl | None<br>**Note:** Under some circumstances, a CEI repository may be required to maintain transaction details. |
| Parlay X Address List Manager over XCAP | None | None |
| WAP Push over SMPP | WAPPushDb2.ddl<br>AdminWAPPushDB2.ddl<br>WAPPushOra.ddl<br>AdminWAPPushOra.ddl | WAPPUSHSENDDATA<br>WAPPUSHNOTIFICATIONDATA |
| Parlay X SMS over SMPP | SmsDbDb2.ddl<br>AdminSmsSmppDb2.ddl<br>SmsDbOra.ddl<br>AdminSmsSmppOra.ddl | SMSSENDDATA<br>SMSRECEIVEDATA<br>SMSNOTIFICATIONDATA<br>SMSOBJECTDATA |
| Parlay X Terminal Location over MLP | TermLocMLPDbDb2.ddl<br>AdminLocMLPDb2.ddl<br>TermLocMLPDbOra.ddl<br>AdminLocMLPOra.ddl | TLMLPNOTIFYDATA |
| Parlay X Multimedia Messaging over MM7 | AdminMmsMm7Db2.ddl<br>MmsAttachmentDb2.ddl<br>MmsDataDb2.ddl<br>AdminMmsMm7Ora.ddl<br>MmsAttachmentOra.ddl<br>MmsDataOra.ddl | MMSATTACHCONTENT<br>MMSATTACHMETADATA<br>MMSNOTIFICATIONDATA<br>MMSSENDDATA<br>MMSRECEIVEDATA |
| Parlay X Terminal Status over Parlay | AdminTsParlayDb2<br>TermStatDbDb2.ddl<br>AdminTsOra.ddl<br>TermStatDbOra.ddl | TERMINALSTATUSNOTIFICATIO<br>TERMINALSTATUSPENDINGNOT |

Table 22. Database tables for Web services (continued)

| Web service | DDLs | Database tables created or updated |
|---|---|---|
| Parlay X Terminal Location over Parlay | AdminTlDb2.ddl<br>TermLocDbDb2.ddl<br>AdminTlOra.ddl<br>TermLocDbOra.ddl | TERMINALLOCATIONNOTIFICATIO |
| Parlay X SMS over Parlay | SmsDbDb2.ddl<br>AdminSmsParlayDb2.ddl<br>SmsDbOra.ddl<br>AdminSmsParlayOra.ddl | SMSSENDDATA<br>SMSRECEIVEDATA<br>SMSNOTIFICATIONDATA<br>SMSOBJECTDATA |
| Parlay X Call Handling over Parlay | CallHandlingParlayDbDb2.ddl<br>AdminChParlayDb2.ddl<br>CallHandlingParlayDbOra.ddl<br>AdminChParlayOra.ddl | CHACCEPTLIST<br>CHRULE<br>CHBLOCKLIST<br>CHCONDITIONALFORWARD<br>CHNOTIFICATION |
| Parlay X Call Notification over Parlay | CallNotificationParlayDbDb2.ddl<br>AdminCnParlayDb2.ddl<br>CallNotificationParlayDbOra.ddl<br>AdminCnParlayOra.ddl | CNNOTIFICATIONINFO<br>CNCALLEVENTS<br>CNTARGETINFO |
| Parlay X Third Party Call over Parlay | AdminTpcParlayDbDb2.ddl<br>AdminTpcParlayDbOra.ddl<br>ThirdPartyCallParlayDbDb2.ddl<br>ThirdPartyCallParlayDbOra.ddl | TPCDATA<br>THIRDPARTYCALLSESSION |
| Address Masking component Web service | AddressMaskingDbDb2.ddl<br>AddressMaskingDbOra.ddl | TRANSACTIONIDENTIFIER<br>INPUTVALUE<br>OUTPUTVALUE<br>REQUESTER<br>INTERNALEXPIRY<br>EXTERNALEXPIRY<br>EXECUTECOUNT |

Table 23. Stored procedures for Web services

| Web service | DDLs | Stored procedures created |
|---|---|---|
| Parlay X Call Handling over Parlay | CallHandlingStoredProcDb2.ddl<br>CallHandlingStoredProcOra.ddl | LOADRULE<br>SAVERULE<br>TESTLOAD |
| Parlay X Third Party Call over Parlay | ThirdPartyCallParlayDbDb2.ddl<br>ThirdPartyCallParlayDbOra.ddl<br>**Note:** This DDL file creates both database and tables for the Web service. Therefore, you need to execute it only once. | DELETETPCRECORDS |

## Planning for security

WebSphere Telecom Web Services Server uses the standard WebSphere Application Server Network Deployment authorization mechanisms and also supports the use of the Trust Association Interceptor. Additional authorization capabilities are provided using Service Policy Manager, which is policy based.

### Security features

Telecom Web Services Server provides the following features:

- The ability to provide granular authorization down to the level of the invoked operation. Authorization for execution of a particular operation is driven by policy information and is a simple filter indicating whether or not a given invocation is allowed to execute. Authorization policy can be defined at different levels in the policy hierarchy: per-requester, per-service, and per-operation. This hierarchy is resolved by the Service Policy Manager during policy retrieval, returning the most specific authorization value. If the requester is not authorized to invoke a particular service or operation, the request is rejected. This requirement is met by the Service Authorization mediation primitive. For additional information on Service Authorization mediation primitives, see the topic *Service Authorization*.
- Providing a means of persisting all transaction information and message contents for integrity purposes. This data should be stored within a relational database for external access. This information can be fed into security tools or reconciled later for other accounting purposes.
- Support for the Trust Association Interceptor security component. This is the recommended security mechanism for Telecom Web Services Server.
- Support for integration with external security systems, such as network monitoring and operations systems.

## General considerations for setting up security

Refer to these general considerations to achieve the most efficient and secure use of Telecom Web Services Server.

When planning security for Telecom Web Services Server, consider the following:

- Telecom Web Services Server supports WebSphere Application Server global security for the application server and administrative console. Global security means that administrative and application security is enabled. Roles defined for Telecom Web Services Server applications must be mapped to users or groups when turning on security. For more information, refer to the WebSphere Application Server Information Center.
- The Trust Association Interceptor (TAI) security component is the recommended security mechanism for handling traffic from the IMS network. A single sign-on scheme is recommended for transactions between the Access Gateway and the Service Platform components, and optionally between the Access Gateway and the Service Policy Manager.

  **Note:** This model requires a two cluster deployment. If the TAI is not used, and a combined Access Gateway and Service Platform components deployment is used, then the Web service implementations will run with security enabled. This requires roles for mapping to all authenticated users. Calls can be enabled directly to the Web service implementations from the enterprise domain, .bypassing the Access Gateway. This method is not ideal, unless you take steps to prevent direct calls. You can do this by instituting a firewall.
- Java 2 security must be turned off.
- JMS/SIBus security must be turned off.
- HTTP digest authentication is supported.

**Note:** When WebSphere Application Server global security is disabled, the Telecom Web Services Server user is assigned to the ″unauthenticated″ user.

## Common user registry

If you enable security but do not choose to use the TAI, then you must use an LDAP directory server, such as IBM Directory Server, to define a common user registry for the Service Platform components, the Access Gateway, and the Service Policy Manager.

The Access Gateway acts as a point of authentication, so it requires a user registry. For authentication to work properly, the following are strongly recommended:

- The Access Gateway must share a common user registry with the Service Platform components.
- Single sign-on certificates must be established between the Access Gateway server and the server that houses the Service Platform components. Refer to the WebSphere Application Server information center for information about configuring certificates for single sign-on.

## Users and user roles

The following users are required when you deploy WebSphere Telecom Web Services Server.

- A user for access to the database, such as db2inst1 for DB2. For Oracle, a user needs to be created as described in the installation section of this Information Center. The database user needs to be defined in the operating system and the database server.
- A user for Service Policy Manager. All authenticated users can be granted policy retrieval access. For policy administration, the role should be mapped to an administrative user.
- A WebSphere Application Server Administrator, which should also be granted the TWSSAdministrator role. In addition, the user should be added to the Integrated Solutions Console users list. This user is defined in the WebSphere Application Server user registry.
- A user for managing service policies. This user is granted the PolicyAdministrator role and is defined in the WebSphere Application Server user registry. This user can be a new user or one that previously existed. When deploying of Service Policy Manager, you should map the PolicyAccessor role to this user.
- A user for each service implementation. Examples of such users include `TS_Parlay_TSAccessor` and `SMS_SMPP_SMSAccessor`. The users can be mapped to all authenticated users or to a group of users that are enrolled in the system. To enroll each user in the system, the user needs to be added to the user registry and have policies set up. Each user is defined in the WebSphere Application Server user registry.

The Telecom Web Services Server services require that requesters be authenticated. All of the users you define will be authenticated at the Access Gateway before being passed to the individual services.

## Considerations for the Access Gateway

The Access Gateway supports standard WebSphere Application Server security features. Access Gateway supports Web services security capabilities according to the level supported by the underlying WebSphere ESB and WebSphere Application Server. For a secure environment, application and administration security must be enabled. Web services security is intended to be applied between third-parties and Access Gateway exported flows. The WebSphere Enterprise Service Bus information center has instructions for enabling Web services security.

The Access Gateway acts as an authentication point for Web service access to the IMS network. The assumption is that the Web services reside in the trusted network. When the Access Gateway passes requests to Web services, it includes header information in the HTTP request. The Access Gateway must be able to front converged HTTP and SIP Web service requests for SIP service implementations. This will require support for the session affinity model.

WebSphere ESB version 6.1.0.2 does not support propagation of session information. The Access Gateway provides an extension to the platform to allow HTTP information to be propagated. This propagation capability is only a conduit. Access Gateway instances are intended to be stateless, which assures that any failover considerations are pushed to the requesting client application.

The Web service implementations run with security disabled, and they use asserted identities to pass security credentials between the secured Access Gateway and the trusted Web services. A trusted network is assumed; however, you can enable transport-level security if desired.

Single sign-on certificates are used for transactions between the Access Gateway and the Service Platform components. This makes it possible for the Access Gateway to pass requests to the Service Platform components without authentication credentials, which is the recommended approach.

### Considerations for Service Platform components

Service Platform components typically are deployed using the TAI trust mechanism, with transport layer security using IPSec or SSL.

Service Platform components also support transport level security (HTTP digest authentication). You can enable SSL (HTTPS) authentication on the HTTP server.

### Considerations for Web service implementations

In addition to the security plans you implement for Telecom Web Services Server, the Web service implementations can also be affected by the way in which security is configured for the remote network elements with which they interact–for example, IBM XDMS or the Parlay Connector.

Some of the Web service implementations require specific Telecom Web Services Server policies if you plan to enable security. For more information, review the policy descriptions for each service implementation you plan to deploy. The descriptions are found in the topic *Reference: default policies for the Access Gateway and Web services*.

## User groups and security roles

You can map security roles to users or groups during deployment.

Roles are initially mapped to groups of users so that if specific users change, the role mapping will not need to change. These groups should be defined prior to running the configuration process. You can then change the group by adding or deleting users, without having to change the initial configuration.

*Table 24. Group names and role mappings*

| Group name | Role mapping | Description |
|---|---|---|
| PolicyAdminGroup | PolicyAdministrator | Administrator of the policies |
| All authenticated users | PolicyAccessor | Anyone accessing policy values |
| All authenticated users | All other Web services interface roles | Clients of the Web services |
| NotifyAdminGroup | NotifyManagementAdministratorRole | Administrator of the notifications |
| TWSSAdminGroup | TWSS Administration Console Administrator, Parlay Administrator, WEST Administrator | Administrator of the TWSS Administration Console or Parlay Administration console |

The First Steps script user interface and response files can be used to configure the group names.

## Planning authentication security using the Trust Association Interceptor

The Trust Association Interceptor (TAI) security component is intended to enhance the overall authentication security for the IBM WebSphere software for Telecom. Implementation scenarios describe how the TAI interacts with the Access Gateway and Service Platform components.

When setting up the Trust Association Interceptor for Telecom Web Services Server, consider the following:

- Install the TAI on the same server as the Service Platform components. The Service Platform uses a trusted network security model using the TAI.
- Avoid installing the TAI on the Access Gateway server unless you have a way of ensuring that none of the incoming messages require authentication–for example, a firewall is in place, or the Access Gateway is called only from inside a trusted network. Note that the Access Gateway uses a single sign-on scheme for transactions with the Service Platform components and optionally with the Service Policy Manager.
- The TAI is used by other components, for example IBM XDMS and Presence Server.

### About the scenarios

The following sections depict common system configurations in which components are deployed in production and in scaling/high-availability (HA) scenarios. Each scenario is presented with the following conditions:

- The scenario does not illustrate all of the possible valid combinations of the IBM WebSphere software for Telecom.
- WebSphere Application Server Administrative node deployment is not shown. It is assumed that all components described in this section belonging in a cluster also belong to a WebSphere Application Server-administered core group for purposes of administration and management.

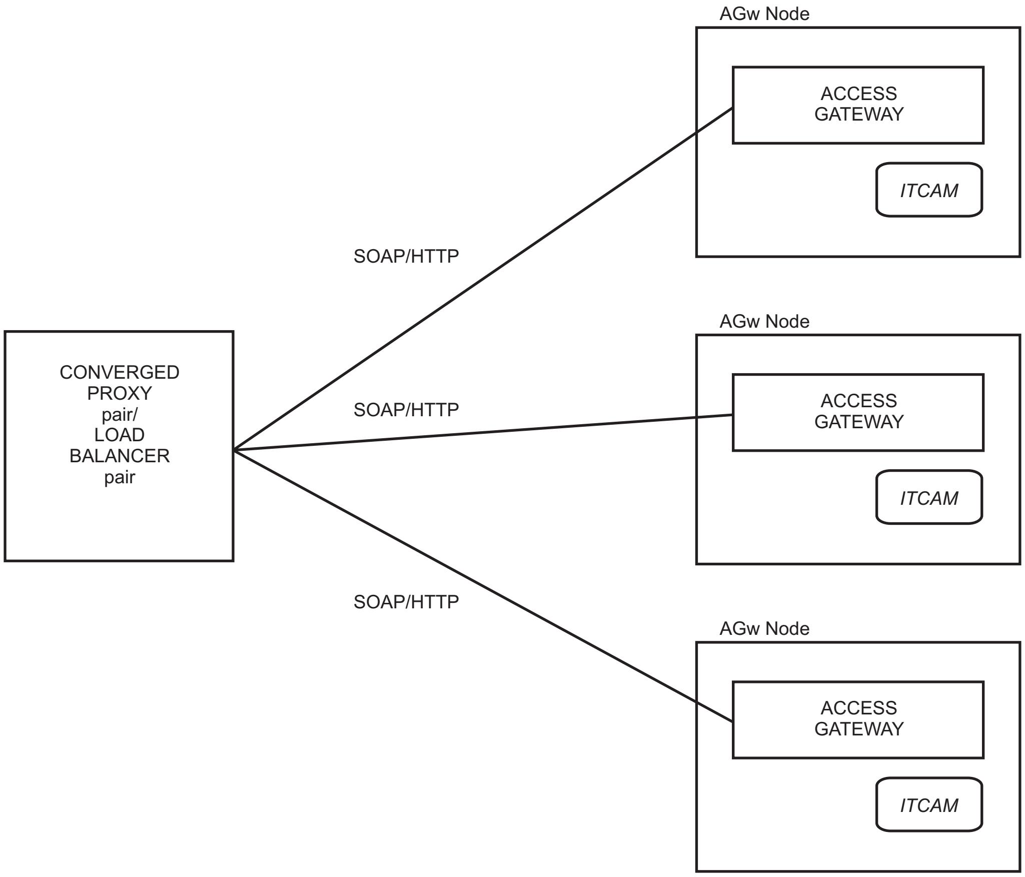
- While the scenario depicts the standard deployment of the IBM Tivoli Composite Application Management (ITCAM) for J2EE operations component that is co-located with the IBM WebSphere software for Telecom, the required remote ITCAM for J2EE components (or other SNMP-based monitors) are not shown. In all cases for interaction with the ITCAM for J2EE operations performance Servlet, it is expected that the cluster load balancer or proxy is not invoked to route requests.
- Session data replication details are not shown.
- The configuration does not address high-availability in an end-to-end sense, nor does it address interactions with required databases.
- Issues of development-to-deployment using the IBM WebSphere Telecom Toolkit are not addressed.
- The example illustrates the components and nodes in a cluster, in the WebSphere Application Server scaling sense consisting of two or more nodes providing identical service. These nodes may or may not cooperate at some level to guarantee high availability. Three nodes are shown as a convention, but other configurations are possible.

  **Note:** The Network Deployment high availability (HA) schemes require an even number of nodes in order to support pair-wise replication

The Telecom Web Services Server constituent components are deployed and scaled separately from each other. There are two cluster descriptions for the Telecom Web Services Server configuration:
- Access Gateway
- Service Platform components

## Access Gateway

The following diagram illustrates a scenario in which three Access Gateway nodes receive SOAP/HTTP traffic that flows through a converged proxy pair or a load balancer pair.



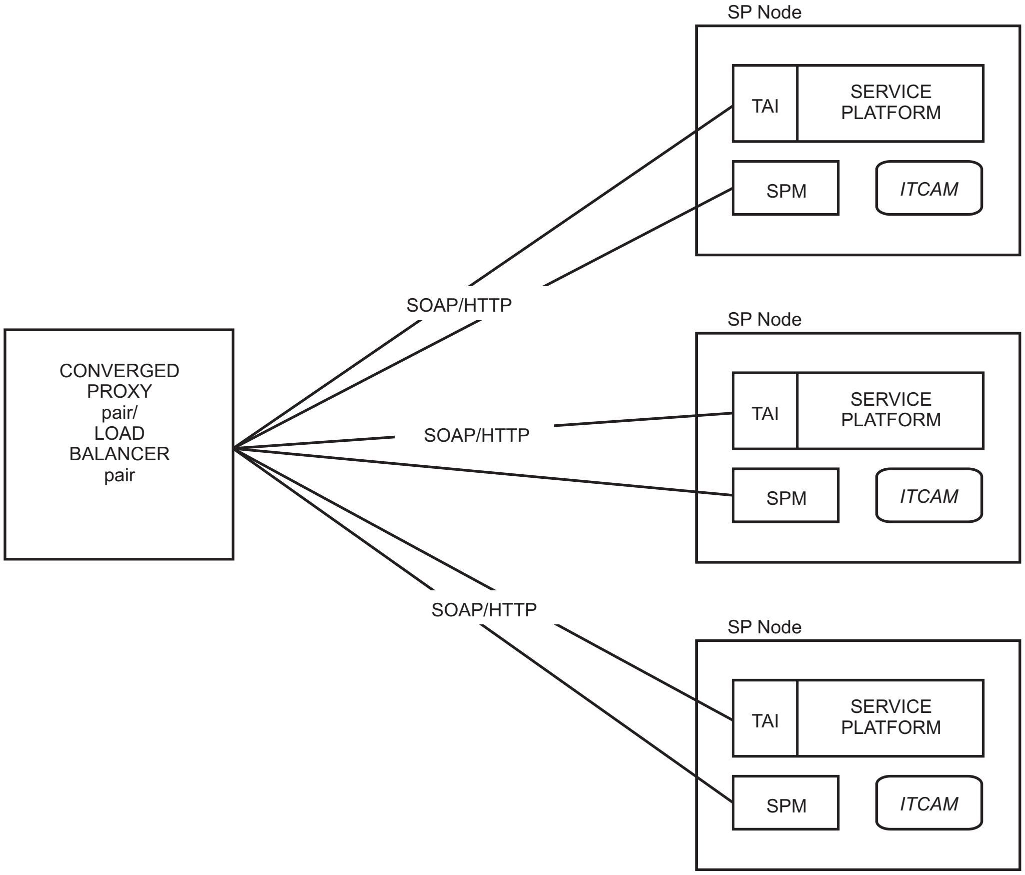
**Note:**

- The WebSphere Application Server converged proxy or any third-party load balancer can be deployed pair-wise (for high availability).
- IBM Tivoli Composite Application Management (ITCAM) users communicate directly with the Access Gateway node for purposes of collecting Performance Monitoring Infrastructure (PMI) data from that node.
- If the converged proxy is also used in the overall Telecom Web Services Server configuration, then the same instance could also be used for HTTP traffic through the Access Gateway; otherwise, IBM HTTP Server (IHS) or another load balancer can be used.
- The Access Gateway performs authentication. Requests hit the Access Gateway directly from the Internet, without a reverse proxy security server (RPSS) being invoked. The Access Gateway can also accept requests from other Service Platform components, where it maintains and propagates the asserted identity header that is present.
- The Access Gateway adds an X-3GPP-asserted identity header to the HTTP request.

## Service Platform components

It is recommended that the Service Policy Manager (SPM) be co-deployed with the Service Platform components, but it may also be deployed in a standalone cluster as well. If deployed in a standalone cluster, the configuration is similar to that for the Access Gateway cluster. The service implementation functionality can be mixed and matched on the service platform using normal WebSphere Application Server application deployment.

The following diagram illustrates a scenario in which three Service Platform nodes, with the Trust Association Interceptor deployed on each one, receive SOAP/HTTP traffic that flows through a converged proxy pair or a load balancer pair.



**Note:**

- The Converged Proxy must be used if there are SIP services deployed; otherwise, IHS or another load balancer can be used
- Only the Access Gateway supports Performance Monitoring Infrastructure (PMI) utilized by the IBM Tivoli Composite Application Management (ITCAM) for J2EE operations component.
- The Trust Association Interceptor is in front of the Service Platform and searches for the asserted identity header.
- The Service Policy Manager uses a single sign-on scheme for transactions with other Telecom Web Services Server components. It can use the Trust Association Interceptor for transactions with the network, but this is not required.

# Planning for notification

As you use the Telecom Web Services Server (TWSS) APIs to develop services, there are a number of ways in which you can set up notification for your applications. Three basic notification options are available to you, and you can use them individually or in combination to get the notification scheme that best fits your needs.

## Overall considerations

When planning the notification setup that makes the most sense for the services you are developing, consider the following factors:

- In order for you to use the interfaces described in this topic, the service you are creating must expose a notification WSDL interface.
- If you are creating a Parlay X-compliant Web service, consider the following: Use the XNotification interface for sending inbound notifications to the application, and use the XNotificationManager interface to register and deregister notifications. Both of the requests are received on the Parlay X Management interface. For example, the SMSNotificationManager interface is an example of a Parlay X Management interface.Notification Management component Web service to register and deregister notifications as these requests are received on the Parlay X management interface. (An example of a Parlay X management interface is the SMSNotificationManager interface.)
- If the service you are creating is not Parlay X-compliant, you cannot use the Px Notify interface for outbound notification. However, you still can use the Notification Management component Web service to register and deregister notifications and to track statistics on received inbound notifications.

The TWSS product provides the following two notification interfaces as Service Platform components:

Notification Management
PX Notification

Most of the Parlay X specifications define a Notification interface. For example, the SMSNotificationManager defines a Notification interface.

You can use these interfaces in combination, and you can use them individually, as needed.

Each option is described in more detail in the following sections.

## Notification Management component Web service

The Notification Management component is designed to provide simple information that can be used by an administrator or care representative to view information about active notifications in the system. The API also allows those individuals to reset outstanding notifications through termination; upon termination, the client-side application would need to reestablish the notification.

Provided with the TWSS product as one of the Service Platform components, Notification Management captures information about active notification registrations on the Service Platform. You can use this information for administrative purposes. Web service implementations register and remove notifications as they are created or torn down. Notification Management also gathers statistics about notification deliveries and failures. These statistics can provide useful information for an administrator wishing to gain insight into notification delivery in a system while it is running.

This component is intended only for capturing information about notifications on the platform and for capturing statistics around notification delivery attempts. Each Web service implementation should manage storage of notification data in the manner that is most efficient for it. The Notification Management component includes an administrative interface for querying notification information and for terminating active notifications.

Notification Management is used by the SMSNotificationManager service implementation to register created notifications or to remove a registration for stopped notifications. Optionally, a custom Web service implementation might also use this component, if there is a need for administrative tracking of created notifications.

Notification Management is also used by the PX Notification component to track statistics about sent notifications. Although there is no user interface for managing these registered notifications, the information can be accessed through a Web services API.

For more information, refer to the topic Notification Management component Web service in the Introduction section of this information center.

## PX Notification component Web service

Like Notification Management, PX Notification is provided with the TWSS product as one of the Service Platform components. It works with the Parlay X-compliant Web service implementations, providing facilities for delivering notifications through the Access Gateway to the destination endpoint.

PX Notification is used to format and process outbound notifications when they are received from the network. For example, the SMS Web service invokes PX Notification for outbound messaging flows and for outbound delivery receipt flows (that is, flows from the telecom network, through TWSS, toward the application).

PX Notification invokes the Notification Management component in order to track statistics–for example, how many notifications of a given registered type have been sent. For a custom service, this call by PX Notification to Notification Management is effectively a no-op if there are no tracked registered notifications in the Notification Management common component (that is, if the custom service did not originally use Notification Management to register created notifications).

Note that each Parlay X interface has a corresponding WSDL and interface for PX Notification. For example, SMSNotification is the specific version of PX Notification that must be used within SMS service implementations.

Many Parlay X Web service implementations require sending outbound Web Service notifications to Web service clients as an indicator of network events. A single network event might result in multiple outbound notifications. The PX Notification component provides facilities for delivering notifications to the destination endpoint through the TWSS Access Gateway. This component also provides an asynchronous model for invocation, in which an application can send a notification, continue its processing without blocking, and then receive a confirmation of delivery. PX Notification is specific to Parlay X because it contains an interface design intended to deliver Parlay X notifications and because its implementation need to include Parlay X-specific stubs.

PX Notification decouples the Web service implementations from the mechanics of delivering the notifications. This includes any necessary persistence of notifications and modification of outbound messages so that they can pass through the Access Gateway.

### SMSNotificationManager interface

The SMSNotificationManager interface, defined as part of the Parlay X standard, is the external interface used by a TWSS application to specify which SMS notifications the application is to receive.

The SMS-based Web service implementations use the Notification Management component Web service to track such notification registrations.

## Planning to migrate from the previous version of Telecom Web Services Server

For users of version 6.2 of the Telecom Web Services Server (TWSS) product, there are several points to consider when migrating your existing configuration to version 7.0.

The TWSS version 7.0 program code should be installed only on a deployment manager or single server instance. Although previous releases of the product were installed uniquely to each node in the deployment, this is no longer necessary. However, when you begin the process of migrating to version 7.0, you will need to uninstall the previous version from each node.

The First Steps configuration script in version 7.0 supports several deployment topologies. Generally speaking, you will use the First Steps script on each deployment manager node where TWSS version 7.0 is installed.

It is possible to transfer workloads from old (version 6.2) Web service implementations to the new ones that are provided in version 7.0. For example, you can set up a parallel service platform with the new Web service implementations while the Access Gateway and Service Policy Manager remain at the version 6.2 level. The interfaces between the Access Gateway and the Service Policy Manager are compatible and will work across versions.

The section *Migrating from a previous release of Telecom Web Services Server*, in this information center, describes the process of migrating your TWSS installation to version 7.0 on both a test system and a production system.

If you used the previous version of TWSS to develop applications, your applications will run with the version 7.0 product. However, new features have been introduced and some interface details have changed for version 7.0. For more information, see the topic *Migrating your applications from a previous release of IBM WebSphere Telecom Web Services Server*, in the Developing section of this information center.

### Coexistence among components

Different versions of the TWSS components can coexist as follows.

- **Access Gateway**: A version 7.0 instance of the Access Gateway can communicate with version 6.2 instances of the Service Platform components and Web service implementations. In such a scenario, however, the Access Gateway cannot make use of the new features that are provided as part of the Service Platform components, such as address masking. Also, keep in mind that different versions of the Access Gateway cannot coexist within the same cluster.
- **Service Platform**: Different versions of the Service Platform components can coexist within a server or a cluster. For example, a version 7.0 instance of the Notification Management component Web service can coexist with version 6.2 instances of all of the Service Platform components.
- **Service Policy Manager**: The version 7.0 Service Policy Manager can coexist with a version 6.2 Service Policy Manager in the same cluster. It is not necessary to update policies to run with the version 7.0 Service Policy Manager, although you should consider updating your policies to take advantage of new features in TWSS version 7.0.
- **TWSS Administration Console**: The console can integrate with the other TWSS components at either the version 7.0 or version 6.2 level. The version 7.0 console can integrate with WebSphere Application Server version 7.0.0.1 or version 6.1.0.x.
- **Web service implementations**: Version 7.0 instances of the Web service implementations can operate together with version 6.2 instances, except as follows:
  - A version 7.0 instance of WAP Push cannot interoperate with a version 6.2 instance of SMS over SMPP. Similarly, a version 7.0 instance of SMS over SMPP cannot interoperate with a version 6.2 instance of WAP Push. (Version 7.0 instances of the two services can interoperate when their **service_type** settings are configured for co-deployment. Refer to the *Administering* portion of this information center for details.)
  - A version 6.2 instance of Terminal Status over Parlay cannot exist on the same server with a version 7.0 instance of Terminal Status over Parlay or on the same server with any instance of WAP Push over SMPP or SMS over SMPP.
  - A version 6.2 instance of Terminal Location over Parlay cannot exist on the same server with a version 7.0 instance of Terminal Location over Parlay or on the same server with any instance of WAP Push over SMPP or SMS over SMPP.

## Considerations for upgrading the TWSS base components

As you plan to upgrade the TWSS base components, consider the following.

- It is not necessary to retrieve and preserve runtime data when you are upgrading the Access Gateway and Service Policy Manager components to version 7.0. As a result, the upgrade steps for these components primarily involve stopping the server, replacing the program code, and restarting the server.
- If you use customized message processing flows for the Access Gateway, some modifications might be required. Changes for TWSS version 7.0 include:
  - In the default message processing flow for version 7.0 of the Access Gateway, the Message Logging mediation primitive is replaced with a Message Interceptor mediation primitive. (You can still use the Message Logging mediation primitive in your custom flows.) If you want to migrate to the Message Interceptor mediation primitive while preserving data that was collected by the old Message Logging mediation primitive, you will need to

  copy data to a new database table. For details, see the *Message Interceptor* topic in the Mediation Primitives section of this information center.
  - The Service Level Agreement (SLA) mediation primitives have been consolidated so that all processing is now done at the cluster level. You can continue to use the SLA Local Enforcement mediation primitive; however, it does not support a time window longer than one second. For details, see the *SLA Enforcement* topic in the Mediation Primitives section of this information center.
- System configuration data for the Service Policy Manager is maintained in the form of policies. Take the following things into account as you migrate to TWSS version 7.0:
  - If any of the policy names change, or if your deployment assumptions (configuration options) change, you must migrate the policy data manually to a new database.
  - Some deployment options in the WebSphere Integrated Solutions Console are tied to specific software versions, which necessitates manual migration steps.
  - New default policies are introduced in version 7.0 for some of the Web service implementations. For a complete list see *Considerations for upgrading the Web service implementations*, elsewhere in this topic.
- There are no special considerations for upgrading the Service Platform components. However, you may want to modify your applications to take advantage of the Address Masking component Web service–new in TWSS version 7.0.
- There are no special considerations for upgrading the TWSS Administration Console for TWSS version 7.0. The upgraded console can integrate with WebSphere Application Server version 7.0.0.1 or version 6.1.0.x.

## Considerations for upgrading the Web service implementations

The following sequence of steps is recommended for upgrading a Web Service implementation in a clustered environment without disruption:

1. Defederate one of the active nodes.
2. Upgrade the node to the new Web service implementation, either by updating it or by uninstalling the old version and installing the new one. During this time, the other active nodes can continue to process requests.

   **Note:** If you use a distributed database configuration, refer to the topic *Migrating data in a distributed database configuration* when you performing the updates.
3. Federate the node on which you upgraded the Web service implementation.
4. Repeat steps 1 through 3 for each remaining node.

(For details on how to perform these steps, refer to the instructions in the section *Migrating from a previous release of Telecom Web Services Server*.)

Some of the Web service implementations entail special considerations when you upgrade them to run in a version 7.0 environment. Those considerations are listed here.

- Migration of runtime data is not supported. However, if you want to recreate notifications that exist from your TWSS version 6.2 installation, you can create a new database and then manually copy the contents of the following database tables:
  - SMSNOTIFICATIONDATA for SMS over SMPP

  - TLMLPNOTIFYDATA for Terminal Location over MLP
  - MMSNOTIFICATIONDATA for MMS over MM7
- For information about migrating applications that are based on the Web service implementations, refer to the topic *Migrating your applications from a previous release of Telecom Web Services Server*. The following additional considerations apply to all of the Parlay-based Web service implementations:
  - The scheduler classes and EJBs (com.ibm.wast.parlay.scheduler.*) for the Parlay Connector are deprecated in version 7.0.
  - The QueuedObjectInvoker class is no longer used in version 7.0. For example, it is replaced in the Cluster communication utility JAR file by the RemoteCaller class.
  - Applications that use plain objects as callbacks must use stateless session beans as their callbacks.
- There are changes to the database tables and schema for version 7.0. In a shared database configuration, run the First Steps script with the **Migrate** option to perform the necessary updates.

  Database schema changes apply to the following Web service implementations: Parlay X Third Party Call over SIP/IMS, WAP Push over SMPP, Parlay X SMS over SMPP, and Parlay X Multimedia Messaging over MM7. For details, refer to the topic *Changes to database tables for Telecom Web Services Server version 7.0*.
- New default policies are introduced in version 7.0 for the following Web service implementations. (All default policies used in TWSS version 7.0 are listed in the TWSS Reference section of this information center.)

  **Parlay X Presence over SIP/IMS**

  service.config.presence.supplier.PublishPresenceTimeout
  service.config.presence.supplier.SubscribePresenceWinfoTimeout
  service.config.presence.supplier.MaximumPublishDurationAllowed

  **Parlay X Terminal Status over SIP/IMS**

  service.config. ProcessUpToLimit
  service.config. MaximumTargets

  **Parlay X Address List Manager over XCAP**

  service.config.UserName

  **WAP Push over SMPP**

  service.config.messaging.Billing
  service.custom.messaging.ServiceType

  **Parlay X SMS over SMPP**

  service.config.messaging.Billing
  service.custom.messaging.ServiceType

  **Parlay X Terminal Location over MLP**

  service.config.ProcessUpToLimit

## Worksheets

Before beginning the installation, you can use these blank worksheets to record your decisions about the locations of network components.

## Component planning worksheet

Use this worksheet to record your decisions about how Telecom Web Services Server (TWSS) components will be distributed in your network.

The following table lists the components of TWSS. You can print the table and use the space provided to record information about the server where each component is deployed. Refer to this information during the installation and configuration process. The first line is filled as an example.

| **Component** | **Server host name (fully-qualified)** | **Node name** | **Cluster name** | **Cell name** |
|---|---|---|---|---|
| *Access Gateway* | *mydomain.example.com* | *node A* | *Access Gateway* | *cell01* |
| Access Gateway | | | | |
| Service Policy Manager (runtime) | | | | |
| Service Policy Manager (console) | | | | |
| TWSS Administration Console | | | | |
| Parlay X Call Notification over SIP/IMS | | | | |
| Parlay X Presence over SIP/IMS | | | | |
| Parlay X Terminal Status over SIP/IMS | | | | |
| Parlay X Third Party Call over SIP/IMS | | | | |
| Parlay X Payment (PostPaid) | | | | |
| Parlay X Address List Manager over XCAP | | | | |
| WAP Push over SMPP | | | | |
| Parlay X SMS over SMPP | | | | |
| Parlay X Terminal Location over MLP | | | | |

| Component | Server host name (fully-qualified) | Node name | Cluster name | Cell name |
|---|---|---|---|---|
| Parlay X Multimedia Messaging over MM7 | | | | |
| Parlay X Terminal Status over Parlay | | | | |
| Parlay X Terminal Location over Parlay | | | | |
| Parlay X SMS over Parlay | | | | |
| Parlay X Call Handling over Parlay | | | | |
| Parlay X Call Notification over Parlay | | | | |
| Parlay X Third Party Call over Parlay | | | | |
| Address Masking component Web service | | | | |

## Distributed database planning worksheet

If you use a distributed database configuration, you can use this sheet to record your decisions about how the database tables will be distributed in the network.

If you use the standard process for installing and configuring Telecom Web Services Server, the scripts will assign default values that will meet your needs in most cases. However, you can use this worksheet if you require a non-default configuration.

The table lists all of the components that require database tables. The first line is provided an example. Use the table to store the database server host name (fully-qualified), port, database name, database alias, data source, and DDL file name. Reference this information during the installation and configuration process.

| Component | Database server host name | Database name | Port | Database Alias | Data source | Data source JNDI name | DDL file name |
|---|---|---|---|---|---|---|---|
| *Access Gateway* | *mydomain.example.com* | *AGDB* | *50000* | *AgDb* | *AgDataSource* | *jdbc/ AGDB* | *ESBDbDb2.ddl* |
| Access Gateway | | | | | | | |

| Component | Database server host name | Database name | Port | Database Alias | Data source | Data source JNDI name | DDL file name |
|---|---|---|---|---|---|---|---|
| Service Policy Manager (runtime) | | | | | | | |
| Service Policy Manager (console) | | | | | | | |
| Notification Management component Web service | | | | | | | |
| Address Masking component Web service | | | | | | | |
| Usage Record component Web service | | | | | | | |
| TWSS Administration Console | | | | | | | |
| Parlay X Call Notification over SIP/IMS | | | | | | | |
| Parlay X Presence over SIP/IMS | | | | | | | |
| Parlay X Terminal Status over SIP/IMS | | | | | | | |
| Parlay X Third Party Call over SIP/IMS | | | | | | | |
| Parlay X Payment (PostPaid) | | | | | | | |

| Component | Database server host name | Database name | Port | Database Alias | Data source | Data source JNDI name | DDL file name |
|---|---|---|---|---|---|---|---|
| Parlay X Address List Manager over XCAP | | | | | | | |
| WAP Push over SMPP | | | | | | | |
| Parlay X SMS over SMPP | | | | | | | |
| Parlay X Terminal Location over MLP | | | | | | | |
| Parlay X Multimedia Messaging over MM7 | | | | | | | |
| Parlay X Terminal Status over Parlay | | | | | | | |
| Parlay X Terminal Location over Parlay | | | | | | | |
| Parlay X SMS over Parlay | | | | | | | |
| Parlay X Call Handling over Parlay | | | | | | | |
| Parlay X Call Notification over Parlay | | | | | | | |
| Parlay X Third Party Call over Parlay | | | | | | | |

# Chapter 3. Installing WebSphere Telecom Web Services Server

WebSphere Telecom Web Services Server (TWSS) has multiple components that should be distributed across multiple servers in the network.

It is common, but not required, to install the Access Gateway in a different cluster from the Service Platform components and the Web service implementations.

The Service Policy Manager can be installed in the same cluster with the Service Platform components and the Web service implementations, or in a separate cluster of its own.

Most components require database tables for storing data. A single database server can be used for all the components, but for performance and failover, the database tables can be distributed across multiple servers. The components include:

- Access Gateway: Provides default message processing flows to define how calls flow to the applications and from the applications back out. The Access Gateway must be installed on the server where you have WebSphere Enterprise Service Bus installed.
- Service Policy Manager: Provides a way to manage policies for the Service Platform components and Web service implementations. The installation instructions assume that you are installing Service Policy Manager on the same node where you installed theWebSphere Enterprise Service Bus.
- Service Platform components and Web service implementations: These are deployed as enterprise applications on WebSphere Application Server. The Service Platform components are used by Web service implementations and must be deployed on the same server. The Web service implementations provide additional functionality, and you can deploy only those that are needed in your network.

For many installations, it is appropriate to run with all of the TWSS components on a single cluster. Others will find it more effective to deploy the components on separate clusters. This information center includes installation scenarios for both cases.

The instructions in this section assume that you are configuring the database tables on a dedicated database server. The First Steps script will create the necessary tables for the selected set of services in a single database instance. If a distributed database instance is desired, database DDL files and scripts are provided to assist with the table creation.

## Telecom network integration scenarios

The overall goal of the Telecom Web Services Server (TWSS) installation is to deploy the Web service implementations that are needed in your network. Use the scenarios as examples to help you understand the installation process.

On a system running WebSphere Enterprise Service Bus version 6.1.0.2, you can install all of the TWSS components on a single cluster. In this case, the Access Gateway, Service Policy Manager, and Service Platform components are installed together.

You can also install the TWSS components on multiple clusters. This is required, for example, when you are running with WebSphere Application Server version 7.0.0.1. The Service Policy Manager, and Service Platform components can run with WebSphere Application Server version 7.0.0.1 or 6.1.0.21. However, the Access Gateway requires WebSphere ESB version 6.1.0.2, which is not compatible with WebSphere Application Server version 7.0.0.1.

## Installing Telecom Web Services Server on a single cluster

In this scenario, the Access Gateway, the Service Policy Manager, and the Service Platform components are deployed together on the same cluster.

### About this task

The single cluster installation scenario is valid only when the embedded WebSphere Application Server Network Deployment and WebSphere ESB are both running version 6.1.0.x.

This scenario guides you through the steps on how to set up a single-clustered environment.

**Note:** This deployment is recommended for most installations. You will only need to run the installation scripts on the deployment manager.

1. Review the prerequisites information to ensure that you have the correct operating system, hardware, and software needed to complete the installation.
2. Configure the operating system for each server where you plan to install Telecom Web Services Server (TWSS) components.
3. Install the database server software on the database server node. Optionally, install the database client on the deployment manager node.
4. Install the WebSphere Enterprise Service Bus (ESB) product on each server.

   The WebSphere Application Server product is embedded within WebSphere ESB. Make sure that WebSphere Application Server is at the version level that is required to run the TWSS Service Platform components.
5. Create the following three groups in the user registry:
   - TWSSAdminGroup
   - PolicyAdminGroup
   - NotifyAdminGroup

   **Note:** The group names are used in the First Steps script.
   a. You start with a clean profile, by using the WebSphere Application Server *manageProfiles* command to delete the old profile, and then use the profile management tool to create a new profile.
   b. Create the cluster. You can give the cluster a descriptive name, such as *TWSScluster*.

      **Note:** Make sure to use the ESB profile for the Access Gateway on ESB.
6. Use the Telecom Web Services Server base installer to install the TWSS base components. There are two installation procedures available: Interactive installer and Silent.
7. Use the TWSS Services installer to install one or more Web service implementations. There are two installation procedures available: Interactive installer and Silent.
8. Run the First Steps script to configure your system.

The First Steps script configures WebSphere, Access Gateway, Service Policy Manager, Service Platform components, and Service implementations. This is based on the user input provided by the First Steps script on the configuration panel or response file.

The First Steps script also publishes necessary changes and software to each server instance in the cluster. The First Steps script also deploys any application EARs that are selected in the installation.

9. If necessary, use the TWSS Administration Console to perform additional configuration for your Web services.

   **Note:** Connections to the backend network elements–such as SMPP, MLP, Parlay gateway, and MM7 servers–generally need to be configured after the system is started. You can perform this configuration using the TWSS Administration Console. Additionally, the Parlay Administration Console can be used for the parlay gateway configuration.

## Installing Telecom Web Services Server on multiple clusters

Some installations might require a more complex environment for deploying the Telecom Web Services Server product (TWSS) . The Access Gateway, the Service Policy Manager, and the Service Platform components are installed on separate clusters.

### About this task

This scenario guides you through setting up an environment to deploy the TWSS components on separate clusters:

- Cluster for the Access Gateway
- Cluster for the Service Policy Manager and the Service Platform components

**Note:** The Telecom Web Services Server installers and First Steps script have to run for each deployment manager that is used. If each cluster has a unique deployment manager, the Telecom Web Services Server installers and the First Steps script will run separately. If all of the clusters are controlled by a single deployment manager, then the Telecom Web Services Server installer and the First Steps script will only need to be run once.

It is important to note that the First Steps script deploys and configures all of the base components and services that are installed on the server where it runs. Configurations in which a single server is shared between multiple clusters, and a different set of services is deployed on each cluster, are not supported.

Other installation configurations are possible, and it is beyond the scope of this document to detail all of the possible options. However, you can use this scenario to extrapolate a configuration that works for your network.

1. Review the prerequisites information to ensure that you have the correct operating system, hardware, and software needed to complete the installation.
2. Configure the operating system for each server where you plan to install Telecom Web Services Server (TWSS) components.
3. Install the database server software on the database server node. Optionally, install the database client on the deployment manager node.
4. Install the prerequisite software.

   a. Install WebSphere Application Server version 7.0.0.1 or version 6.1.0.21 on the clusters where you install the Service Policy Manager and Service Platform components.
   b. Install WebSphere Enterprise Service Bus version 6.1.0.2 (which includes WebSphere Application Server) on the Access Gateway cluster.

   **Note:** If you are running WebSphere Application Server version 6.1.0.21, the TWSS Access Gateway can coexist in the same cluster with the Service Platform components. However, when the Service Platform components are deployed in a cluster running WebSphere Application Server version 7.0.0.1, the Access Gateway must be deployed in a separate cluster.

5. Create the following three groups in the user registry:
   - TWSSAdminGroup
   - PolicyAdminGroup
   - NotifyAdminGroup

   **Note:** The group names are used in the First Steps script panel:
6. Create the clusters for the Access Gateway and the Service Platform components. (This scenario assumes that the Service Policy Manager and all of the Web service implementations are in a single cluster. However, they can be in different clusters if needed.)
   a. Create the Access Gateway cluster. Assign a descriptive name to the cluster, such as *AccessGateway*.
   b. Create the cluster for the Service Platform components. Assign a descriptive name to the cluster, such as *Services*.
7. Use the Telecom Web Services Server base installer to install the Access Gateway on the *AccessGateway* cluster. There are two installation procedures available: Interactive installer and Silent.
8. Use the Telecom Web Services Server base installer to install the following components on the *Services* cluster. There are two installation procedures available: Interactive installer and Silent.
   - Service Policy Manager
   - Service Platform components
   - TWSS Administration Console
   - Parlay Administration Console (if needed)
   - First Steps configuration script
9. Use the TWSS Services installer to install the required service implementations on the *Services* cluster. In addition, the First Steps script will need to be installed in each cluster. There are two installation procedures available: Interactive installer and Silent.

   The services installer will have the required database support files and the connector support component for a variety of services. This includes services such as the Parlay and SMPP connectors.
10. From the database server, create and initialize the database using crtsrvDb2 or crtsrvOra. This is provided as part of the TWSS base installation.

    The create-database script (`crtsrv*`) is not visible on the Feature installation panel. It is installed only if you select the features that require them. For more details, refer to the topic *Customizing the installation response file for Telecom Web Services Server base components*.

11. Run the First Steps script on the Access Gateway cluster to configure your system. The wizard performs the following tasks:
    - Configure common resources for WebSphere Application Server
    - Configure common resources for the Access Gateway
    - Process all configurations for each server instance (syncServers)
12. Run the First Steps script on the Services cluster to configure your system. The wizard configures the common resources for WebSphere Application Server.
13. If necessary, use the TWSS Administration Console to perform additional configuration for your Web services.

    **Note:** Connections to the backend network elements–such as SMPP, MLP, Parlay gateway, and MM7 servers–generally need to be configured after the system is started. You can perform this configuration using the TWSS Administration Console. Additionally, the Parlay Administration Console can be used for the parlay gateway configuration.

## Migrating from the previous version of Telecom Web Services Server

For users of Telecom Web Services Server (TWSS), the topics in this section describe the process of migrating your existing version 6.2 configuration to the new 7.0 version.

Topics in this section describe the migration for both your test (non-production) systems and your production systems. Additional topics describe special steps when performing the migration in a clustered environment and when you are using a distributed database configuration. A final topic describes the steps to take to finish the migration.

Before embarking on the migration, you should review the information in the topic *Planning to migrate from the previous version of Telecom Web Services Server*.

### Migrating a non-production system to Telecom Web Services Server version 7.0

To migrate a non-production or test system from Telecom Web Services Server (TWSS) version 6.2 to version 7.0, begin by uninstalling the previous version of the product. Then install and configure the new version.

#### Before you begin

Become familiar with the planning information provided in the topic *Planning to migrate from a previous release of Telecom Web Services Server*.

#### About this task

When migrating an existing system to Telecom Web Services Server version 7.0, the program code is updated independently of your configuration data. Not all of the existing configuration data will necessarily apply to the new version 7.0 system. However, wherever possible, your existing configuration data can be brought forward to the new release.

The following steps outline the process for migrating a non-production, or test, system to version 7.0.

1. Verify the hardware topology. For details, refer to the topic *Standard configurations for Telecom Web Services Server*.

   In TWSS version 7.0 it is recommended–but no longer required–that you run the Access Gateway in a separate cluster from the other product components. As a result, you may choose to modify your clustering topology to use TWSS Version 7.0.
2. Verify that the appropriate version of WebSphere Application Server and all other prerequisite software is installed. For details, refer to the topic *Software requirements*.
3. Stop all application servers, node agents, and deployment managers. For details, refer to the topic *Stopping and starting the server*.
4. Remove applications and uninstall the program code for the old version of Telecom Web Services Server. For detailed instructions, refer to the *Removing Telecom Web Services Server* section of the information center for the product version you are replacing.

   The First Steps script will migrate data only from a consolidated or shared database, not from a distributed database. Verify that the Service Policy Manager tables, CONFIGPROPERTIES and CFGPROPERTIES (if Parlay is used), tables exist in the primary database before proceeding.
5. Run the TWSS base installer to install version 7.0 program code on the new deployment manager, or on the single server instance if there is no deployment manager. For details, see the *Installing Telecom Web Services Server base components* section of this information center.
6. Run the TWSS services installer to install version 7.0 program code for the Web service implementations on the new deployment manager, or on the single server instance if there is no deployment manager. For details, see the *Installing Telecom Web Services Server services* section of this information center.
7. Start the deployment managers and all node agents. For details, refer to the topic *Stopping and starting the server*.
8. Before migrating the usage record data in DB2, perform the following:
   a. Create a bufferpool using the command: `CREATE BUFFERPOOL USAGEDB32KBP all nodes SIZE 10000 AUTOMATIC PAGESIZE 32K;`
   b. Create a tablespace using the command: `CREATE TABLESPACE USAGEDB32KTS PAGESIZE 32K MANAGED BY AUTOMATIC STORAGE EXTENTSIZE 256 PREFETCHSIZE 64 BUFFERPOOL USAGEDB32KBP NO FILE SYSTEM CACHING;`
   c. Export the usage record data from the TWSS 6.2 table.
   d. Import the usage record data exported from the TWSS 6.2 table into the tablespace created in step b.
9. Run the First Steps script to configure each deployment manager or server on which you installed the TWSS program code. The First Steps script has a Migrate option that enables you to migrate old policies and configuration data.

   In a distributed database configuration where there is one database per cluster, run the First Steps script, using the Migrate option, on each cluster. In each case, make sure that the database parameters refer to the database that pertains to that cluster–for example, AGDB for the Access Gateway cluster or SPMDB for the Service Policy Manager cluster.

   In a distributed database configuration where there is more than one database per cluster, create a new temporary database and run the First Steps script on each cluster. You will need to migrate the distributed database instances manually by doing the following:

   a. Run the database setup script (`crtsrvxx`) for each database instance, specifying the appropriate migration DDL as the input parameter. The names of the migration DDLs are listed in the topic *Changes to database tables for Telecom Web Services Server version 7.0*.
   b. Configure the data sources in the WebSphere Integrated Solutions Console to point to each database.

   For details, refer to the topic *Migrating data in a distributed database configuration*.
10. Create additional data sources, as necessary, and update the necessary JNDI bindings to the respective data sources.
11. Test the database connection and start the clusters or server instances.

## Migrating a production system to Telecom Web Services Server version 7.0

To migrate a production system from Telecom Web Services Server (TWSS) version 6.2 to version 7.0, you should plan to run two separate systems in parallel. After the first system has been fully migrated, has been tested, and is up and running, you can move the workload from the older running system to the newer running system.

### Before you begin

Become familiar with the planning information provided in the topic *Planning to migrate from a previous release of Telecom Web Services Server*.

### About this task

When migrating an existing system to Telecom Web Services Server version 7.0, the program code is updated independently of your configuration data. Not all of the existing configuration data will necessarily apply to the new version 7.0 system. However, wherever possible, your existing configuration data can be brought forward to the new release.

The following steps outline the process for migrating a production system to TWSS version 7.0.

1. Perform load balancing, or add new capacity, so that the workload for the system to be migrated is running at no more than half of the total capacity. This activity will probably require you to set up new nodes, with a new deployment manager.
2. Verify the hardware topology. For details, refer to the topic *Standard configurations for Telecom Web Services Server*.

   In TWSS version 7.0 it is recommended–but no longer required–that you run the Access Gateway in a separate cluster from the other product components. As a result, you may choose to modify your clustering topology to use TWSS Version 7.0.
3. Stop all application servers and node agents that are no longer needed for running the existing system. These servers and node agents, together with any new hardware that you have set up, are known in this procedure as *migration elements*. They will be used to run TWSS version 7.0 while the previous version is running on the old system.
4. Create a new deployment manager for the migration elements. The new TWSS version 7.0 cluster should have the same fronting proxy or proxies as the old cluster.

5. On each migration element, remove applications and uninstall the program code for the old version of Telecom Web Services Server. For detailed instructions, refer to the *Removing Telecom Web Services Server* section of the information center for the product version you are replacing.
6. Verify that WebSphere Application Server and all other prerequisite software is installed on each migration element. For details, refer to the topic *Software requirements*.
7. In the new cluster, run the TWSS base installer to install version 7.0 program code on the deployment manager, or on the single server instance if there is no deployment manager. The TWSS base installer installs the Access Gateway, the Service Policy Manager, the Service Platform components, and the Parlay Connector.

   For details, see the *Installing Telecom Web Services Server base components* section of this information center.
8. In the new cluster, run the TWSS services installer to install version 7.0 program code for the Web service implementations on the deployment manager, or on the single server instance if there is no deployment manager. For details, see the *Installing Telecom Web Services Server services* section of this information center.
9. In the new cluster, start the deployment managers and all node agents. For details, refer to the topic *Stopping and starting the server*.
10. For a shared database configuration, create a new database in the new cluster, following the procedures in the topic *Preparing for the Telecom Web Services Server database*.

    For a distributed database configuration, create a new database in each cluster–for example, AGDB for the Access Gateway cluster and SPMDB for the Service Policy Manager cluster. If you have multiple databases on each cluster, create a new temporary database and then configure the database appropriately. Refer to the topic *Migrating data in a distributed database configuration*.

    **Note:** For the Service Platform components (but not for the Web service implementations or any other components), you can preserve existing data by pointing to the existing database rather than creating a new database.
11. Before migrating the usage record data in DB2, perform the following:
    a. Create a bufferpool using the command: `CREATE BUFFERPOOL` *`USAGEDB32KBP`* `all nodes SIZE 10000 AUTOMATIC PAGESIZE 32K;`
    b. Create a tablespace using the command: `CREATE TABLESPACE` *`USAGEDB32KTS`* `PAGESIZE 32K MANAGED BY AUTOMATIC STORAGE EXTENTSIZE 256 PREFETCHSIZE 64 BUFFERPOOL` *`USAGEDB32KBP`* `NO FILE SYSTEM CACHING;`
    c. Export the usage record data from the TWSS 6.2 table.
    d. Import the usage record data exported from the TWSS 6.2 table into the tablespace created in step b.
12. Run the First Steps script to configure each deployment manager or server on which you installed the Telecom Web Services Server program code.

    In a distributed database configuration where there is one database per cluster, run the First Steps script on each cluster using the Initial Configuration Mode option (or the Migrate option if you are configuring an existing database). In each case, make sure that the database parameters refer to the database that is specific to that cluster– for example, AGDB for the Access Gateway cluster or SPMDB for the Service Policy Manager cluster.

In a distributed database configuration where there is more than one database per cluster, create a new temporary database and run the First Steps script on each cluster. You will need to migrate the distributed database instances manually by doing the following:

   a. Run the database setup script (`crtsrvxx`) for each database instance, specifying the appropriate migration DDL as the input parameter. The names of the migration DDLs are listed in the topic *Changes to database tables for Telecom Web Services Server version 7.0*.
   b. Configure the data sources in the WebSphere Integrated Solutions Console to point to each database.

   For details, refer to the topic *Migrating data in a distributed database configuration*.

13. Create additional data sources, as necessary, and update the necessary JNDI bindings to the respective data sources.
14. On the proxy server or servers, specify the TWSS version 7.0 cluster as the default cluster, so that the new dialogs will be routed correctly.
15. Test the database connection and start the clusters or server instances.
16. Move the workload from the existing system to the new version 7.0 cluster:
    a. Using the WebSphere Performance Monitoring Infrastructure (PMI), monitor performance for the applications on the TWSS version 6.2 cluster. When the number of sessions reaches zero, or when it reaches a low enough level that you deem acceptable, stop the cluster.
    b. Remove the remaining nodes from the TWSS version 6.2 cluster and add them to the 7.0 cluster. These nodes can be added to the new cluster to provide additional capacity.
    c. Roll out the applications to the newly added nodes in the TWSS version 7.0 cluster.

## Migrating data in a distributed database configuration

The First Steps script takes care of setting up database tables for Telecom Web Services Server (TWSS) when you are using a consolidated or shared database. However, additional steps may be required when you are using a distributed database configuration.

### Before you begin

This procedure is a sub-procedure within the overall migration process of your test (non-production) or production system. Make sure that you have already installed TWSS version 7.0 components and that you have performed the procedures described in one of the following post-installation configuration topics:

- *Creating and configuring the DB2 database server instance*
- *Creating and configuring the Oracle database server instance*

Note that the First Steps script assumes that all nodes in the cluster are at a TWSS version 7.0 level.

### About this task

If a distributed or partitioned topology is used for databases, you will encounter one of the following scenarios:

- **One database per cluster**. For example, you may have the Access Gateway running in one cluster and the Service Platform components running in another cluster, with each cluster having its own database. If this is the case, when you run the First Steps script on the Access Gateway cluster, all of the database parameters should refer to the Access Gateway database (typically named AGDB). Then when you run the script on the Service Platform cluster, the database parameters should refer to the Service Platform database (typically SPMDB). No additional special considerations apply.

- **More than one database per cluster**. For example, you may have a separate database for the WAP Push service on the same cluster with the database for the Service Platform components. If this is the case, use the procedure in this topic to create a temporary database and then configure new databases to work with TWSS version 7.0.

To migrate a distributed database configuration when there is more than one database per cluster, perform the following steps.

1. On one of the nodes in the cluster, create a temporary database by running the First Steps script with the Initial Configuration Mode option pointing to a new temporary database. For detailed information, see the topic *Running the First Steps configuration script*. The First Steps script creates the minimum necessary data sources to support a distributed topology and points to the new single (temporary) database instance.
2. Migrate the existing distributed databases manually to the database schema for TWSS version 7.0 by running the database setup script (`crtsrvxx`) for each database instance, specifying the appropriate migration DDL as the input parameter. For details about how the schema has changed, and for a list of the names of the migration DDLs, refer to the topic *Changes to database tables for Telecom Web Services Server version 7.0*.
3. Create additional data sources, as necessary, and update the necessary JNDI bindings to the respective data sources.
4. Drop the temporary database.
5. Verify the new migrated environment by running any successful service logic test case. Verify databases and logs.

   **Note:** Migration of runtime data is generally not supported, except in the cases of some Direct Connect-based Web service implementations. (For details, refer to the topic *Planning to migrate from the previous version of Telecom Web Services Server*.) In these cases, be sure that you have copied all of the existing data to the new database before deleting the old database and database tables.
6. Verify that the newly populated data coexists with the previous data in cases where migration of runtime data is supported for a given Web service or feature.

   **Note:** When the First Steps script sets up your configuration for running TWSS version 7.0, it does not modify your existing configuration. Therefore, if you made changes to the default configuration for a previous version of TWSS, your changes are preserved during the migration process.

## Changes to database tables for Telecom Web Services Server version 7.0

For several of the Telecom Web Services Server (TWSS) subcomponents, the format of database tables has changed from version 6.2 to version 7.0. The changes are cataloged here.

## Web service implementations

The following schema changes are made in database tables associated with the Web service implementations.

**Third Party Call over SIP/IMS**

The ThirdPartyCallData record database is no longer used in TWSS version 7.0. Use the First Steps script to create a new Usage Record table and a new THIRDPARTYCALL table.

**WAP Push over SMPP**

The WAPPUSHSENDDATA table holds delivery status data. A new column, MSISDN, is added to store target addresses in MSISDN format.

The contents of the MSISDN column cannot be null. After you configure your new database for TWSS version 7.0 and migrate your data from version 6.2, ensure that there are no null entries in this column. The MSISDN column needs to be filled manually.

The receipt of asynchronous notification responses (submit_SM and deliver_SM) for requests that were sent prior to the migration is not supported.

In TWSS version 7.0, the database sequence for WAP Push over SMPP starts at 1001 and increments by 2.

**SMS over SMPP**

The SMSSENDDATA table holds delivery status data. When you run the First Steps script with the Migrate option, a new column, MSISDN, is added to store target addresses in MSISDN format.

The same considerations that are documented for WAP Push over SMPP also apply here.

In TWSS version 7.0, the database sequence for SMS over SMPP starts at 1000 and increments by 2.

**MMS over MM7**

The ATTRIBUTES column has been renamed to ATTRIB in the following database tables: MMSSENDDATA, MMSRECEIVEDATA, MMSNOTIFICATIONDATA.

The ATTACHMENT_CONTENTS table has been renamed as MMSATTACHCONTENT.

## DDLs for migrating databases

The following table lists the DDLs that are provided for migrating databases.

| Component Name | Source Path | DDL Names |
|---|---|---|
| Access Gateway | *esb_root*`/installableApps/TWSS-Base/database` | ESB-TWSS70DBMigrDbDb2.ddl<br>ESB-TWSS70DBMigrDbOra.ddl |
| Parlay Connector | *was_root*`/installableApps/TWSS-Base/database` | PC-TWSS70DBMigrDbDb2.ddl |
| Usage Records | *was_root*`/installableApps/TWSS-Base/database` | UR-TWSS70DBMigrDbDb2.ddl<br>UR-TWSS70DBMigrDbOra.ddl |

| Component Name | Source Path | DDL Names |
|---|---|---|
| WAPPush over SMPP | *`was_root`*`/installableApps/TWSS-Services/database` | WAPPush-TWSS70DBMigrDbDb2.ddl<br>WAPPush-TWSS70DBMigrDbOra.ddl |
| SMS over SMPP | *`was_root`*`/installableApps/TWSS-Services/database` | SMSSmpp-TWSS70DBMigrDbDb2.ddl<br>SMSSmpp-TWSS70DBMigrDbOra.ddl |
| MMS over MM7 | *`was_root`*`/installableApps/TWSS-Services/database` | MMS-TWSS70DBMigDbDb2.ddl<br>MMS-TWSS70DBMigrDbOra.ddl |

**Note:** If you run the migration DDL scripts to migrate the existing data in the SMSSENDDATA table, the value in the MSISDN column is set to an empty string in the table. Before you start the migration, ensure that there are not any pending *deliver_SM* or *submit_SM* responses for the messages which have already been submitted by TWSS to the SMSC.

**Note:** In the source paths listed in the table:

- *esb_root* represents the WebSphere ESB installation location
- *`was_root`* represents the WebSphere Application Server installation location

## Completing the migration

Use the following information, as needed, to complete the migration from Telecom Web Services Server (TWSS) version 6.2 to version 7.0.

### Migrating service policies

The TWSS service policies are migrated by the First Steps script. You do not need to migrate them manually. However, if you created any custom policies for custom applications, and if you would like to continue using those policies on your version 7.0 system, then you need to migrate those policies manually.

The names of the policies used in TWSS version 6.2 are unchanged in version 7.0. However, the names of some policies used in version 6.1.1 are different. If any of your applications still refer to the version 6.1.1 policy names, refer to the list of new names in the topic *Changed policy names for migration*.

### Migrating runtime data

The database tables related to Access Gateway mediation primitives in TWSS version 7.0 (Transaction Recorder, Network Statistics, and Message Interceptor) contain transient data and do not require migration. If desired, however, you can convert the data and move it to your BSS/OSS system.

### Migrating configuration data

The existing `CONFIGPROPERTIES` table contains configuration data for the Web service implementations. For example, some TWSS Administration Console settings for the Web service implementations contain values that relate to a specific version of software. These can be migrated manually to a new database. Review the settings to determine whether your deployment assumptions have changed from the previous version. The configuration settings are not automated.

If you customized the format of database records in the previous version, you will need to repeat the customization after setting up the databases with the new version.

#### Migrating Parlay applications

Because of changes to the API, it may be necessary to modify Parlay applications that you created for previous versions of Telecom Web Services Server. For details, refer to the topic *Migrating your applications from a previous release of Telecom Web Services Server* in this information center.

## Preparing the environment

A successful installation requires that the prerequisite software be installed and configured. You must prepare the operating systems for the installation, and you must set up the database. Then, the clusters can be created.

Complete the following prerequisite tasks before installing WebSphere Telecom Web Services Server.

### Preparing the operating system

Each operating system has unique configuration needs. You should complete all preparation steps for all prerequisite software before completing the following steps:

#### Preparing an AIX operating system

To prepare your AIX system for installation, you must allocate disk space appropriately and apply certain patches. You must complete these steps in order for the system to function properly with WebSphere Telecom Web Services Server.

##### About this task

Comprehensive information about preparing AIX is found in the WebSphere Application Server Network Deployment information center.

#### Preparing a Linux operating system

To improve performance and prevent network issues, you must run the Name Service Caching Daemon. To ensure your servers are properly synchronized, you should also run the Network Time Protocol daemon.

##### About this task

Comprehensive information about completing the operating system preparation may be found in the WebSphere Application Server Network Deployment information center.

The Name Service Caching Daemon must be started on each Linux server where you plan to deploy WebSphere Telecom Web Services Server components.

1. Use **grep** to determine if the Name Service Caching Daemon is running. For example:

```
ps -elf | grep "nscd"
```

2. If it is not running, start it.
   a. Change to the /etc/init.d directory. For example:
      ```
      cd /etc/init.d
      ```
   b. Run **nscd**. For example:
      ```
      ./nscd start
      ```
3. To configure the Name Service Caching Daemon to start when the system is started, update *rcdefault_runlevel.d*.
   a. To determine the *default_runlevel*, observe */etc//inittabn*. You will see the following:
      ```
       id:5:initdefault:
       # System initialization.
       si::sysinit:/etc/rc.d/rc.sysinit
      ```
      The first line indicates that the default runlevel is **5**.
   b. To update *rc5.d*, do the following:
      ```
      cd /etc/rc.d/rc5.d
      ln -s ../init.d/nscd S60nscd
      ln -s ../init.d/nscd K60nscd
      ```
4. Ensure that the Network Time Protocol daemon is configured and running.

## Preparing a Solaris operating system

To prepare your Solaris system for installation, you must allocate disk space appropriately and apply specific patches. You must complete these steps in order for the system to function properly with the WebSphere Telecom Web Services Server.

### About this task

Comprehensive information about completing the operating system preparation may be found in the WebSphere Application Server Network Deployment information center.

1. Install the *en_US* locale language patch.

   **Note:** The product supports *en_US* only.
2. After applying the locale patch, make sure that the locale is set to *en_US*. Use the following commands:
   ```
   export LANG=en_US
   export LC_ALL=en_US
   ```

   Alternatively, you can set the locale permanently for a particular profile.
3. Ensure that common commands are recognizable by the operating system. This is done by adding the `/usr/ucb/` path to the PATH variable:
   ```
   export PATH=$PATH:/usr/ucb/
   ```

   Alternatively, you can set the path permanently for a particular profile.

   **Note:** It is recommended that you add the above mentioned environment variables permanently to the user profile, this will avoid conflict between multiple opened sessions.

   To set the environment variables permanently to a user profile, refer to the topic titled *Setting Environment Variables* in the Solaris Advanced User's Guide. This document is available on the Sun Solaris Web site.

## Preparing for the Telecom Web Services Server database

You will need one or more databases to run WebSphere Telecom Web Services Server. Much of the database setup is done automatically when you follow the standard procedures for installing and configuring the product. However, a few preparatory steps are required.

### About this task

For IBM DB2 Enterprise Server Edition, you create and initialize your database after installing the TWSS base components. The First Steps script–part of the standard configuration procedure–creates the required database tables.

For Oracle Database, you must create the database and the user IDs before installing WebSphere Telecom Web Services Server. Following the installation, the First Steps script creates the required database tables.

If you plan to use a distributed database configuration, begin by setting up a consolidated database using the First Steps script. Then migrate specific tables to other database servers as desired. The instructions in this information center describe the setup for a consolidated database.

The two database configurations, consolidated and distributed, are described in the topic *Planning for the database*.

### Preparing to set up the database on DB2

As you prepare to set up the Telecom Web Services Server database (TWSSDB), ensure that certain DB2 settings are configured correctly.

#### Before you begin

You must have installed and started IBM DB2 Enterprise Server Edition Version 9.5 FixPak 1 on the database server.

#### About this task

Run *crtsrvxxx* to create the database.

1. Log in to the DB2 server with a user ID that has database administrator authority, such as db2inst1.
2. Add your WebSphere administrator user ID to the DB2 group (usually `db2grp1`). You will use this administrator ID later, when you create and configure your database (refer to the topic *Creating and configuring the DB2 database server instance*).
3. Optional: Adjust the logging level in the `SystemOut.log` to Severe or higher. This is necessary to suppress warning messages (J2CA0294W) that would otherwise display because direct JNDI lookup was deprecated in WebSphere Application Server Version 6.0.

   For example:

   ```
   *=info:com.ibm.ejs.j2c.ConnectionFactoryBuilderImpl=severe
   ```

   For details, refer to the topic *Log level settings* in the WebSphere Application Server 7.0 Information Center.

## Preparing to set up the database on Oracle

As you prepare to set up the Telecom Web Services Server database (TWSSDB), ensure that certain Oracle settings are configured correctly. You must also create a database user

### Before you begin

You must have completed the following steps:

- Installed and started a supported version of Oracle Database on the database server

### About this task

For WebSphere Telecom Web Services Server, you must create a database user. In this information center, the name twssuser is used; however, this is only an example and you can use another name. The user IDs must be created before creating the database tables. If you are using multiple database servers, you must create the user on each server.

A separate database for the Parlay Connector database is not required. If you plan to deploy any of the Parlay-based Web service implementations, the Parlay Connector can use database tables in TWSSDB.

Complete the following steps on the database server where you want to create the database. The provided script does not create the database. The script only adds new tables to an existing database. During the procedure you will need the Oracle SYSTEM user ID.

1. Use the Oracle Database Configuration Assistant utility to create the database that WebSphere Telecom Web Services Server will use. You must create the following databases depending on your database topology:
   - If you are using a consolidated database topology, create one database: TWSSDB.
   - If you are using a distributed database topology, create the following databases (these names are recommended) :

     TWSSDB
     AGDB
     ATTACHDB
     SPMDB
     ADMINDB
     USAGEDB
     PARLAYDB
     SLADB
     MESSAGEDB
2. Optionally, adjust the logging level in the `SystemOut.log` to **Severe** or higher. This is necessary to suppress warning messages **(J2CA0294W)**, that would otherwise display because direct JNDI lookup was deprecated in WebSphere Application Server version 6.0.

   For example:

   ```
   *=info:com.ibm.ejs.j2c.ConnectionFactoryBuilderImpl=severe
   ```

### What to do next

Later, after you install the TWSS base components and the Web service implementations, you will configure the Oracle database using the `crtsrvOra.sh` script. Refer to the topic *Creating and configuring the Oracle database server instance*.

## Creating the clusters

Telecom Web Services Server makes use of the horizontal and vertical clustering capabilities that are offered with WebSphere Application Server. Using various clustering techniques, you can achieve a high-availability system.

Before installing the Telecom Web Services Server product, you must create one or more clusters. There are a variety of ways in which to do this:
- You can create a single cluster for all of the Telecom Web Services Server components.
- You can create two or more clusters: for example, a cluster for the Access Gateway and another cluster for the nodes where you will deploy the Service Platform components and Web service implementations. (The topic *Standard configurations for Telecom Web Services Server* illustrates a typical setup with two clusters.)
- You can set up a single deployment manager, with unique clusters for each of the components but sharing the same nodes.

This information center provides instructions for setting up the following typical configuration:
- One database server
- One cluster for the Access Gateway
- One cluster for the Service Policy Manager
- One cluster for Service Platform components and Web service implementations

You must create an HTTP proxy server (and optionally a SIP proxy server) and associate the proxy server or servers with each cluster that you create. Refer to the WebSphere Application Server information center for additional information about setting up the proxy server. A link is provided at the end of this topic.

### Creating the cluster for the Access Gateway

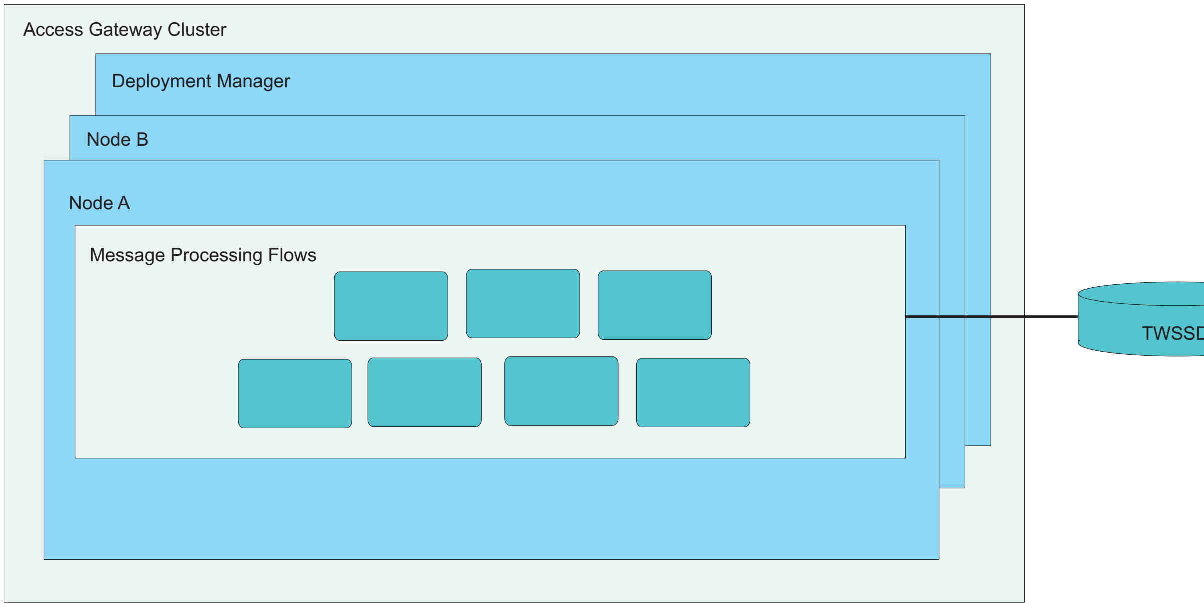
The Access Gateway normally resides in a separate cluster from the TWSS Service Platform components. When setting up the cluster for the Access Gateway, you first install the deployment manager. Then you install the standalone servers and add them to the deployment manager. Finally, before you install the WebSphere Telecom Web Services Server components, you create the clusters.

#### Before you begin

You must have completed the following steps:
- Installed and set up the database
- Obtained WebSphere Enterprise Service Bus, Version 6.1.0.2

#### About this task

A simple Access Gateway cluster is illustrated in the following diagram. The message processing flows are deployed on one or more nodes in the cluster, and they exchange data with the shared database (TWSSDB).



Use the *ESBProfile* for the WebSphere Application Server Network Deployment and WebSphere Enterprise Service Bus. If you need to use an advanced profile, do not select the **for development** profile. It may cause problems with the WebSphere Application Server SIP container.

You must use the WebSphere Enterprise Service Bus tools for creating new profiles and application servers.

When you install a fix pack, you can check the log that is generated during the installation to verify that the fix pack was installed successfully. If the installation was successful, you should see `INSTCONFSUCCESS` as the last entry in the log. If you select the default installation path during the installation of WebSphere Enterprise Service Bus, the logs are stored in the following directory:

- AIX `/usr/ibm/WebSphere/ESB/logs/update/`
- Linux `/opt/ibm/WebSphere/ESB/logs/update/`
- `/opt/ibm/WebSphere/ESB/logs/update/`

To set up a clustered configuration, you must configure cell-managed server nodes on several servers, with the deployment manager on its own server. Refer to the WebSphere ESB Information Center for additional information about this installation scenario.

1. On the server where you plan to install the deployment manager, install the following prerequisite software:
   a. Install WebSphere Enterprise Service Bus version 6.1. If you use the launchpad installer, the WebSphere ESB installation will install the required version of WebSphere Application Server Network Deployment.

      **Important:** Do not attempt to install WebSphere Enterprise Service Bus to a directory path that includes National Language Characters. If the path contains National Language Characters, the installation will not complete.
   b. Install WebSphere Enterprise Service Bus version 6.1.0.2 (Fix Pack 2), using the detailed installation instructions provided with the fix pack.

2. Create a deployment manager profile using the WebSphere ESB Profile creation wizard:
   1. Switch to the directory *esb_root*`/firststeps/esb`, where *esb_root* represents the installation location of the WebSphere ESB.
   2. Issue the following command to start the WebSphere First steps console: `./firststeps.sh`
   3. From the console, launch the Profile creation wizard.

   Refer to the WebSphere ESB information center for additional information about creating a deployment manager profile.
3. Start the deployment manager using either the First Steps script for Dmgr01 or the ./startManager.sh command.
4. On each server where you plan to install application servers, install the following prerequisite software:
   1. Install WebSphere Enterprise Service Bus version 6.1. If you use the launchpad installer, the WebSphere ESB installation will install the required version of WebSphere Application Server Network Deployment.

      **Important:** Do not attempt to install WebSphere Enterprise Service Bus to a directory path that includes National Language Characters. If the path contains National Language Characters, the installation will not complete.
   2. Install WebSphere Enterprise Service Bus version 6.1.0.2 (Fix Pack 2), using the detailed installation instructions provided with the fix pack.
5. Before you can add the application servers to the deployment manager, you may need to update the hosts file on each physical server. The following steps must be completed on each server where you have created application servers.
   1. Open the hosts file in a text editor. The hosts file is in the following directory: `/etc/hosts`
   2. Verify that there is a unique entry for the IP address of the server. By default there is a localhost entry, such as:

      ```
      127.0.0.1  host_name.domain.com host_name localhost.localdomain localhost
      ```

      Replace the default entry with the following line:

      ```
      127.0.0.1  localhost.localdomain localhost
      ```
   3. Then add a new line that specifies the IP address of the server.

      ```
      ip_address

      host_name.domain.com host_name
      ```

      Where:

      *ip_address* is the IP address of the server.

      *host_name* is the host name of the server.

      *domain* is the domain of the server.

      For example:

      ```
      192.0.2.21 testbox.example.com testbox
      ```
   4. Save and close the hosts file.
   5. Repeat this step for each application server.
6. Create a custom profile on each application server node using the WebSphere ESB Profile Creation wizard:

   a. Switch to the directory `esb_root/firststeps/esb`, where *esb_root* represents the installation location of the WebSphere ESB.
   b. Issue the following command to start the WebSphere First steps console: `./firststeps.sh`
   c. From the console, launch the Profile creation wizard.

   Refer to the WebSphere ESB information center for additional information about creating a custom profile.

7. Use the addNode command to add the application server to the cell.
   a. On the physical server that you want to add to the deployment manager, change to the bin directory for the application server profile. For example:

      ```
      cd was_profile_root/bin
      ```

   b. To add a node, run the addNode command. For example:

      ```
      ./addNode.sh dmgr.domain.com
      ```

      Where:

      *dmgr.domain.com* is the fully qualified host name of the deployment manager server.
   c. To verify that the application servers were added to the cell, review the addNode.log. When you run the addNode command, a message displays in the location where the addNode.log is stored, for example: `was_profile_root/AppSrv01/logs`.
   d. Repeat this step for each application server that will be added to the cell.
8. Create the cluster using the Integrated Solutions Console.

   When creating new application servers, select **defaultESBServer** (the Default Server template for WebSphere ESB). Do not select **default** (the WebSphere Default Server template).

   The detailed procedure is found in the topic *Creating a cluster* in the WebSphere ESB information center.
9. Start the deployment manager, nodes, and servers:
   a. Start the deployment manager. Run the following command:

      AIX `was_profile_root/bin/startManager.sh`

      Linux `was_profile_root/bin/startManager.sh`

      `was_profile_root/bin/startManager.sh`

      Where:

      The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).
   b. Start the nodes. Run the following command:

      AIX `was_profile_root/bin/startNode.sh`

      Linux `was_profile_root/bin/startNode.sh`

      `was_profile_root/bin/startNode.sh`

      Where:

      The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).
   c. Start the servers. Run the following command:

      AIX `was_profile_root/bin/startServer.sh server_name -username user_name -password password`

      Linux `was_profile_root/bin/startServer.sh server_name -username user_name -password password`

```
was_profile_root/bin/startServer.sh server_name -username
user_name -password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of the application server profile (for example, AppSrv01).

*server_name* is name of the application server.

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

10. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
    b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    c. Click **Log in**.
11. Start the cluster:
    a. In the navigation pane, click **Servers** → **Clusters**.
    b. Select the **check box** for the cluster.
    c. Click **Start**.
12. Create an HTTP proxy server and associate the proxy server with the cluster that you have created. Refer to the WebSphere ESB Information Center for additional information on setting up the proxy server.

## Creating the cluster for the Service Policy Manager

Creating a separate cluster for the Service Policy Manager is optional.

### Before you begin

You must have completed the following steps:

- Installed and configured the database server
- Created the deployment manager and application servers

### About this task

The Service Policy Manager cluster is created on WebSphere Application Server Network Deployment, version 6.1.0.x or 7.0.0.1. It can share the TWSSDB database instance with the Access Gateway cluster if desired.

If possible, use the default profiles for WebSphere Application Server Network Deployment. Also, avoid customizing the profile with the Profile Management

Tool. If you need to use an advanced profile, do not select the **for development** profile. Doing so can cause problems with the WebSphere Application Server SIP container.

After you have installed the prerequisite software, you must federate the application server nodes into the deployment manager. Then, you can create the cluster using the Integrated Solutions Console. You can create the first cluster member from one of the application servers you federated into the deployment manager. After you have created each application server, you must add each application server to the network deployment manager cell. Additional cluster members can be added using the Integrated Solutions Console at a later time, as needed.

1. On the physical server where you plan to install the deployment manager, install WebSphere Application Server version 7.0.0.1 or 6.1.0.21, using the launchpad. When prompted to select the WebSphere Application Server environments, select **Cell (deployment manager and a managed node)**.

   Refer to the WebSphere Application Server information center for detailed information about the installation.
2. On each physical server where you plan to install an application server, install WebSphere Application Server version 7.0.0.1 or 6.1.0.21, using the launchpad. When prompted to select the WebSphere Application Server environments, select **Application server**.

   Refer to the WebSphere Application Server information center for detailed information about the installation.
3. Before you can add the application servers to the deployment manager, you may need to update the hosts file on each physical server.
   a. Click **Security** → **Global security** and enable both administrative security and application security.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Security** → **Secure administration, applications, and infrastructure**.

      Refer to the WebSphere Application Server information center for detailed information about the installation.
   b. Open the hosts file in a text editor. The hosts file is in the `/etc/hosts` directory.
   c. Verify that there is a unique entry for the IP address of the server. By default there is a localhost entry, for example:

      ```
      127.0.0.1  host_name.
      domain.com host_name localhost.localdomain
      localhost
      ```

      Replace the default entry with the following line:

      ```
      127.0.0.1  localhost.localdomain localhost
      ```
   d. Then add a new line that specifies the IP address of the server.

      ```
      ip_address host_name.domain.com host_name
      ```

      Where:

      *ip_address* is the IP address of the server.

      *host_name* is the host name of the server.

      *domain* is the domain of the server.

      For example:

```
9.32.175.169 testbox.myco.com
testbox
```

   e. Save and close the hosts file.

   f. Repeat this step for each application server.

4. Use the addNode command to add the application server to the cell.

   a. On the physical server that you want to add to the deployment manager, change to the bin directory for the application server profile. For example:

      ```
      cd was_profile_root/bin
      ```

   b. To add a node, run the addNode command. For example:

      ```
      ./addNode.sh dmgr.domain.com
      ```

      Where:

      *dmgr.domain.com* is the fully qualified host name of the deployment manager server.

   c. To verify that the application servers were added to the cell, review the addNode.log. When you run the addNode command, a message displays in the location where the addNode.log is stored, for example: `was_profile_root/AppSrv01/logs`.

   d. Repeat this step for each application server that will be added to the cell.

5. Start the deployment manager, nodes, and servers:

   a. Start the deployment manager. Run the following command:

      AIX `was_profile_root/bin/startManager.sh`

      Linux `was_profile_root/bin/startManager.sh`

      `was_profile_root/bin/startManager.sh`

      Where:

      The `was_profile_root` path contains the name of the deployment manager profile (for example, Dmgr01).

   b. Start the nodes. Run the following command:

      AIX `was_profile_root/bin/startNode.sh`

      Linux `was_profile_root/bin/startNode.sh`

      `was_profile_root/bin/startNode.sh`

      Where:

      The `was_profile_root` path contains the name of a federated node profile (for example, Custom01).

   c. Start the servers. Run the following command:

      AIX `was_profile_root/bin/startServer.sh server_name -username user_name -password password`

      Linux `was_profile_root/bin/startServer.sh server_name -username user_name -password password`

      `was_profile_root/bin/startServer.sh server_name -username user_name -password password`

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

      The `was_profile_root` path contains the name of the application server profile (for example, AppSrv01).

      *server_name* is name of the application server.

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

6. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
7. Federate the application server into the deployment manager cell.
   a. If both server processes are running, use the administrative console for the deployment manager to add the application server node into the cell.
   b. Click **System administration** → **Nodes** → **Add Node**. Then follow the wizard to add the node into the cell. If both processes are on the same machine, you can use `localhost` for the host name.

      **Note:** The SOAP port for the application server node is 8880 unless you changed the port during profile creation.
8. Create the cluster using the Integrated Solutions Console. The detailed procedure is found in the topic *Creating clusters* in the WebSphere Application Server information center.
9. Create an HTTP proxy server and associate it with the cluster that you have created. Refer to the WebSphere Application Server information center for additional information about setting up the proxy server.

## Creating the cluster for Service Platform components and Web service implementations

You can create a separate cluster for the Service Platform components. This cluster will also contain the Web service implementations, which you will install on the same servers as the Service Platform components.

### Before you begin

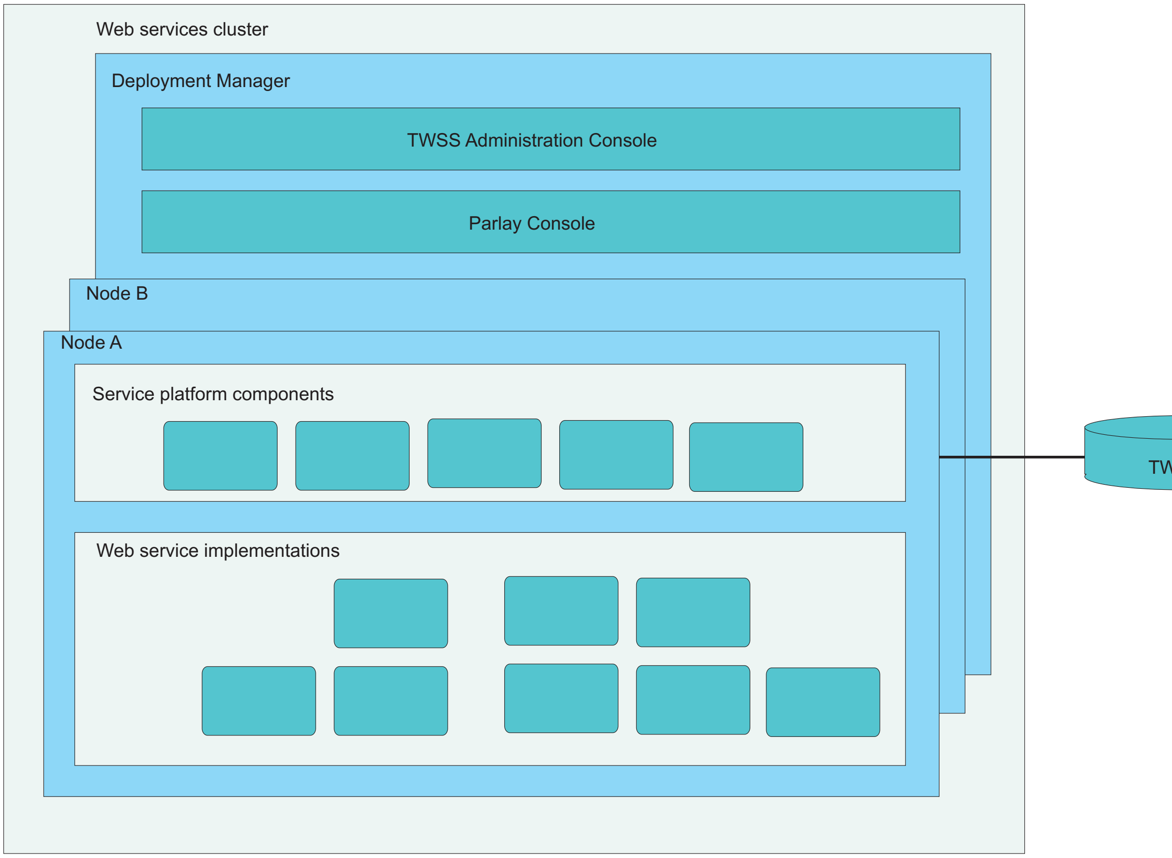
You must have completed the following steps:

- Installed and configured the database server
- Created the deployment manager and application servers

### About this task

The cluster for the Service Platform components and Web service implementations must be created on WebSphere Application Server Network Deployment, version 6.1.0.x or 7.0.0.1.

A simple cluster is illustrated in the following diagram. In the illustration, the Service Platform components are deployed on the same node as the Web service implementations, as required. The TWSS Administration Console is deployed on the deployment manager node, as required. There are unique database instances for theUsage Record component Web service and the Service Platform component configuration data. However, this is not required.


Web services cluster
Deployment Manager
TWSS Administration Console
Parlay Console
Node B
Node A
Service platform components
Web service implementations
TW

If possible, use the default profiles for WebSphere Application Server Network Deployment. Also, avoid customizing the profile with the Profile Management Tool. If you need to use an advanced profile, do not select the **for development** profile. Doing so can cause problems with the WebSphere Application Server SIP container.

After you have installed the prerequisite software, you must federate the application server nodes into the deployment manager. Then, you can create the cluster using the Integrated Solutions Console. You can create the first cluster member from one of the application servers you federated into the deployment manager. After you have created each application server, you must add each application server to the network deployment manager cell. Additional cluster members can be added using the Integrated Solutions Console at a later time, as needed.

1. On the physical server where you plan to install the deployment manager, install WebSphere Application Server version 7.0.0.1 or 6.1.0.21, using the launchpad. When prompted to select the WebSphere Application Server environments, select **Cell (deployment manager and a managed node)**.

   Refer to the WebSphere Application Server information center for detailed information about the installation.

2. On each physical server where you plan to install an application server, install WebSphere Application Server version 7.0.0.1 or 6.1.0.21, using the launchpad. When prompted to select the WebSphere Application Server environments, select **Application server**.

   Refer to the WebSphere Application Server information center for detailed information about the installation.
3. Before you can add the application servers to the deployment manager, you may need to update the hosts file on each physical server.
   1. Click **Security** → **Global security** and enable both administrative security and application security.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Security** → **Secure administration, applications, and infrastructure**.

      Refer to the WebSphere Application Server information center for detailed information about the installation.
   2. Open the hosts file in a text editor. The hosts file is in the `/etc/hosts` directory.
   3. Verify that there is a unique entry for the IP address of the server. By default there is a localhost entry, for example:

      ```
      127.0.0.1  host_name.
      domain.com host_name localhost.localdomain
      localhost
      ```

      Replace the default entry with the following line:

      ```
      127.0.0.1  localhost.localdomain localhost
      ```
   4. Then add a new line that specifies the IP address of the server.

      ```
      ip_address host_name.domain.com host_name
      ```

      Where:

      *ip_address* is the IP address of the server.

      *host_name* is the host name of the server.

      *domain* is the domain of the server.

      For example:

      ```
      9.32.175.169 testbox.myco.com
      testbox
      ```
   5. Save and close the hosts file.
   6. Repeat this step for each application server.
4. Use the addNode command to add the application server to the cell.
   1. On the physical server that you want to add to the deployment manager, change to the bin directory for the application server profile. For example:

      ```
      cd was_profile_root/bin
      ```
   2. To add a node, run the addNode command. For example:

      ```
      ./addNode.sh dmgr.domain.com
      ```

      Where:

      *dmgr.domain.com* is the fully qualified host name of the deployment manager server.

   c. To verify that the application servers were added to the cell, review the addNode.log. When you run the addNode command, a message displays in the location where the addNode.log is stored, for example: *was_profile_root*/AppSrv01/logs.
   d. Repeat this step for each application server that will be added to the cell.
5. Start the deployment manager, nodes, and servers:
   a. Start the deployment manager. Run the following command:

      AIX
      ```
      was_profile_root/bin/startManager.sh
      ```
      Linux
      ```
      was_profile_root/bin/startManager.sh
      ```
      ```
      was_profile_root/bin/startManager.sh
      ```

      Where:

      The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).
   b. Start the nodes. Run the following command:

      AIX
      ```
      was_profile_root/bin/startNode.sh
      ```
      Linux
      ```
      was_profile_root/bin/startNode.sh
      ```
      ```
      was_profile_root/bin/startNode.sh
      ```

      Where:

      The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).
   c. Start the servers. Run the following command:

      AIX
      ```
      was_profile_root/bin/startServer.sh server_name -username
      user_name -password password
      ```
      Linux
      ```
      was_profile_root/bin/startServer.sh server_name -username
      user_name -password password
      ```
      ```
      was_profile_root/bin/startServer.sh server_name -username
      user_name -password password
      ```

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

      The *was_profile_root* path contains the name of the application server profile (for example, AppSrv01).

      *server_name* is name of the application server.

      *user_name* represents your WebSphere Application Server administrator user ID.

      *password* represents the password associated with your *user_name*.
6. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
7. Federate the application server into the deployment manager cell.
   a. If both server processes are running, use the administrative console for the deployment manager to add the application server node into the cell.
   b. Click **System administration** → **Nodes** → **Add Node**. Then follow the wizard to add the node into the cell. If both processes are on the same machine, you can use `localhost` for the host name.

      **Note:** The SOAP port for the application server node is 8880 unless you changed the port during profile creation.
8. Create the cluster using the Integrated Solutions Console. The detailed procedure is found in the topic *Creating clusters* in the WebSphere Application Server information center.
9. Create an HTTP proxy server and associate it with the cluster that you have created. Refer to the WebSphere Application Server information center for additional information about setting up the proxy server.
10. Create a SIP proxy server if you plan to deploy SIP-based Web service implementations, and associate the proxy server with the cluster that you have created.
11. Enable replication for SIP sessions if you plan to deploy SIP-based Web service implementations. Refer to the WebSphere Application Server information center for additional information about replicating SIP sessions.

    **Note:** It is necessary to define user groups in the user registry. Three group names will be requested by the TWSS First Steps configuration script. The group names are configurable; and the following names are recommended:
    - PolicyAdminGroup
    - TWSSAdminGroup
    - NotifyMgmtAdminGroup

## Preparing to use the Trust Association Interceptor

Follow these procedures to install the Trust Association Interceptor (TAI) security component and make it ready for use by the IBM WebSphere Telecom Web Services Server.

The TAI is intended to enhance the overall authentication security for the IBM WebSphere software for Telecom.

It is a good idea to install the TAI on the same server as the Service Platform components. The Service Platform uses a trusted network security model using the TAI.

### Preparing the installation files for the Trust Association Interceptor

Before you install the Trust Association Interceptor (TAI), the WebSphere IMS™ Connector installation file must be unpacked on the servers where WebSphere Application Server is installed.

### Before you begin

Unpacking the WebSphere IMS Connector installation file, `DHAImsConnectorInstallPackage_6.2.0.tar`, which is found on the WebSphere IMS Connector CD, places all of the files for the WebSphere IMS Connector and for the TAI into their appropriate directories.

It is necessary to perform this step only once, even if you plan to use the TAI for several different components.

Install the TAI on the deployment manager and on each node in the Service Platform cluster. Install the TAI on the WebSphere ESB server only if are running the Service Platform on that server. If you are running only the Access Gateway on the ESB server, it is not necessary to install the TAI on that server.

**Note:** *`was_root`* is the installation root directory for WebSphere Application Server Network Deployment. By default, this directory is:

AIX `/usr/IBM/WebSphere/AppServer`

Linux `/opt/IBM/WebSphere/AppServer`

`/opt/IBM/WebSphere/AppServer`

1. On the server where WebSphere Application Server is installed, copy the installation file (`IBM_WebSphere_IMS_Connector/DHAImsConnectorInstallPackage_6.2.0.tar`) from the CD to the *`was_root`* directory.
2. Change (`cd`) to the *`was_root`* directory.
3. Unpack the file by typing the following command: `tar -pxvf DHAImsConnectorInstallPackage_6.2.0.tar`

   **Note:** Remember to apply any relevant fix packs for the TAI.

## Installing the Trust Association Interceptor security component

The Trust Association Interceptor (TAI) is installed from the WebSphere IMS Connector CD onto the server where WebSphere Application Server is installed. After installation, it exists in the IMS trusted domain to intercept HTTP and SIP traffic.

### Before you begin

For a Telecom Web Services Server (TWSS) installation, install the TAI on the server that houses the Service Platform components.

Before installing the TAI, do the following:

- Unpack `DHAImsConnectorInstallPackage_6.2.0.tar` on the server where WebSphere Application Server is installed. (For details, refer to the topic *Preparing the installation files*.)
- Verify that `DHAIMSConnectorTai.jar`, which contains TAI code for both HTTP (HttpInterceptor) and SIP (SipInterceptor), is installed in the directory *`was_root`*`/lib/ext`.

**Note:** *`was_root`* is the installation root directory for WebSphere Application Server Network Deployment. By default, this directory is:

AIX `/usr/IBM/WebSphere/AppServer`

Linux `/opt/IBM/WebSphere/AppServer`

`/opt/IBM/WebSphere/AppServer`

## About this task

Perform the following steps to install the interceptor:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Click **Security** → **Global security** to display the Global security window, and enable both administrative security and application security.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Security** → **Secure administration, applications, and infrastructure**.

   **Note:** Enable single sign-on to recognize traffic from other TWSS components but not traffic that originates from the local host. Make sure that Java 2 security is disabled.

   For detailed instructions about enabling security, refer to the topic *Securing applications and their environment* in the WebSphere Application Server Information Center.
3. Configure general security settings:
   a. In the Global security window, under Authentication, click **Web and SIP security** → **General settings**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Web security** → **General settings**.
   b. Select **Authenticate only when the URI is protected** and **Use available authentication data when an unprotected URI is accessed**.
   c. Click **OK**, then click **Save** to save changes to the master configuration.
4. Configure the interceptor:
   a. In the Global security window, under Authentication, click **Web and SIP security** → **Trust association**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Web security** → **Trust association**.
   b. Click **Enable trust association**.
   c. Under Additional Properties, click **Interceptors**.

   d. Delete the default interceptors by selecting their check boxes and clicking **Delete**. If you need any of the default interceptors, you can add them back after you have added the WebSphere TAI. This ensures that the WebSphere TAI will be invoked first.
   e. Click **New** and type the class name `com.ibm.imsconnector.tai.HttpInterceptor`
   f. Click **Apply**.
   g. Click **OK**, then click **Save** to save changes to the master configuration.
5. Configure custom properties for the HTTP interceptor:
   a. In the Global security window, click **Custom properties**.
   b. Click **New** to add a new custom property.
   c. Define the allowedSenderList property:

      **Name**: allowedSenderList

      **Value**: A comma-delimited list of one or more hosts that the interceptor considers trusted. You can specify host names or IP addresses, and you can use the wildcard character *. For example: `localhost, *@us.example.com, 192.0.2.21`

      **Description** (optional): TAI trusted hosts
   d. Click **Apply**.
   e. Return to the Custom properties window.
   f. Add additional custom properties as needed. For a list of custom properties and their descriptions, refer to the HTTP properties table in the topic *Configuring the Trust Association Interceptor*.
   g. Click **OK**, then click **Save** to save changes to the master configuration.
6. Configure a new SIP interceptor:
   a. In the Global security window, under Authentication, click **Web and SIP security** → **Trust association**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Web security** → **Trust association**.
   b. Under **Additional Properties**, click **Interceptors**.
   c. Click **New** and type the class name `com.ibm.imsconnector.tai.SipInterceptor`
   d. Click **OK**, then click **Save** to save changes to the master configuration.
7. Configure custom properties for the SIP interceptor:
   a. In the Global security window, click **Custom properties**.
   b. Click **New** to add a new custom property.
   c. Define the allowedSenderList property:

      **Name**: allowedSenderList

      **Value**: A comma-delimited list of one or more hosts that the interceptor considers trusted. You can specify host names or IP addresses, and you can use the wildcard character *. For example: `localhost, *@us.example.com, 192.0.2.21`

      **Description** (optional): TAI trusted hosts
   d. Click **Apply**.
   e. Return to the Custom properties window.
   f. Add additional custom properties as needed. For a list of custom properties and their descriptions, refer to the SIP properties table in the topic *Configuring the Trust Association Interceptor*.

   g. Click **OK**, then click **Save** to save changes to the master configuration.
8. Optional: If you require any of the default interceptors that you deleted in step 4 on page 142, add them.
9. Restart the server.

## Installing Telecom Web Services Server base components

Use the Telecom Web Services Server (TWSS) base installer to install the TWSS base components. Later you will run the First Steps script to make your TWSS installation ready for use.

The TWSS base installer installs the following components:
- Service Policy Manager runtime component
- Service Policy Manager Console
- Access Gateway
- Access Gateway applications
- Service Platform components
- Parlay Connector
- TWSS Administration Console
- Parlay Administration Console
- First Steps script (for setting up your basic configuration)
- WSDL

You can run the interactive installer, or you can perform a silent installation.

**Note:** If an earlier version of Telecom Web Services Server is installed, you must uninstall it before proceeding with the installation of version 7.0. Refer to the following sources for more information:
- The topic *Migrating from a previous release of Telecom Web Services Server* in this information center
- The uninstall procedures in the information center for the earlier version

### Installing Telecom Web Services Server base components using the interactive installer

The Telecom Web Services Server (TWSS) base installer guides you through a set of interactive panels to install the TWSS base components.

#### Before you begin

Ensure that the following requirements are met:
- You have installed WebSphere Application Server Network Deployment (version 7.0.0.1 or 6.1.0.21) with a deployment manager and at least one managed node.
- You have federated all nodes into the deployment manager cell.
- You have created a WebSphere user account repository. It is recommended that an LDAP repository be federated with the TWSS base installer.

  **Note:** See the WebSphere Application Server information center for details about performing these tasks.
- You have performed the database preparation steps described in the topic *Preparing for the Telecom Web Services Server database*.

- You have created clusters for the Telecom Web Services Server components you plan to install. Refer to the topic *Creating the clusters* for more information.

The TWSS base installer installs the following features and components:
- Access Gateway
  - Runtime components
  - Access Gateway applications
- Service Platform components
  - Runtime components
  - Parlay Connector
  - TWSS Administration Console
  - TWSS Parlay Administration Console
  - Common Component Applications
  - WSDL
- Service Policy Manager
  - Service Policy Manager runtime component
  - Service Policy Manager console
- First Steps script (for configuring your system after the installation)

Use the TWSS base installer on the WebSphere server. The installer copies the required database DDL files, along with the features selected, to the target server. After installation, the First Steps script will use the copied DDL files to configure the required databases on the database server.

You can simplify the installation by running the Access Gateway and Service Platform components on the same version of the WebSphere Application Server.

The TWSS base installer is organized so that the three major components (Access Gateway, Service Policy Manager, and Service Platform components) can be installed in any order.

To install the TWSS base components, complete the following steps:
1. Stop the servers, nodes, and deployment manager:
   a. Stop the servers. Run the following command:

      AIX
      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      Linux
      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

      The *was_profile_root* path contains the name of the application server profile (for example, AppSrv01).

      *server_name* is name of the application server.

      *user_name* represents your WebSphere Application Server administrator user ID.

      *password* represents the password associated with your *user_name*.

b. Stop the nodes. Run the following command:

AIX
```
was_profile_root/bin/stopNode.sh -username user_name
-password password
```

Linux
```
was_profile_root/bin/stopNode.sh -username user_name
-password password
```

```
was_profile_root/bin/stopNode.sh -username user_name -password
password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of a federated node profile (for example, Custom01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name.*

c. Stop the deployment manager. Run the following command:

AIX
```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

Linux
```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

2. Navigate to the `/TWSS_Install` directory on the CD.
3. Launch the base installation wizard by issuing the following command:
   ```
   ./setup
   ```
   The installation process always creates an install trace file in the /tmp directory `TWSS-base_InstallTrace.log`. This file is helpful in determining any install problems and is required by the product support team for any install-related problems. After the installation is done, this trace file is moved to the *`was_root`*`/logs` directory.

   A separate log file is also created under the *`was_root`*`/logs` directory: `TWSS-base_InstallLog.log`. The log file is translated to user selected language during install.
4. The Welcome panel displays with English as the default. Click **OK**.
5. Click **Next**.
6. Read the license agreement. Then select **I accept the terms in the license agreement**, and click **Next**.
7. Review the information in the Product information panel and click **Next**.
8. Select a WebSphere Application Server instance from the list.
9. Click **Next**.

10. In the Features panel, select the components you want to install, and click **Next**.
11. In the Summary Information panel, verify your selections. Use the **Back** button to go back and make changes. When you are satisfied with your selections, click **Next**.
12. When the installation is complete, click **Finish** to exit the wizard.

### What to do next

To check that the installation was successful, refer to the topic *Verifying the results after installing Telecom Web Services Server base components*.

## Installing Telecom Web Services Server base components silently

The Telecom Web Services Server (TWSS) base components can be installed silently without user interaction, by reading user responses from a response file.

### About this task

A silent installation uses the TWSS base installer to install the product in silent mode, without the graphical user interface. Instead of displaying a script interface, the silent installation causes the installation script to read all of your responses from a response file that you provide.

To specify non-default options during a silent installation, customize the response file. To install silently, you must accept the license agreement using the agreement option within the response file.

### Customizing the installation response file for Telecom Web Services Server base components

Before invoking the Telecom Web Services Server (TWSS) base installer in silent mode, you need to customize the response file that will be used to provide user responses to the installer.

#### Before you begin

Ensure that the following requirements are met:

- You have installed WebSphere Application Server Network Deployment (version 7.0.0.1 or 6.1.0.21) with a deployment manager and at least one managed node.
- You have federated all nodes into the deployment manager cell.
- You have created a WebSphere user account repository. It is recommended that an LDAP repository be federated with the TWSS base installer.

  **Note:** See the WebSphere Application Server information center for details about performing these tasks.
- You have performed the database preparation steps described in the topic *Preparing for the Telecom Web Services Server database*.
- You have created clusters for the Telecom Web Services Server components you plan to install. Refer to the topic *Creating the clusters* for more information.

## About this task

Use the response file to supply values to the TWSS base installer. Running in silent mode, the installer does not display interactive dialogs. Instead, it reads values from the response file.

To modify the install response file, you must copy the sample response file, `twss_base_install.rsp`, from the CD.

**Note:** If the installer fails before or while processing the parameter that specifies the target directory, the log file will be in the system's temporary directory.

To customize the response file in preparation for a silent installation of the TWSS base components, follow these steps:

1. Locate the response file (`twss_base_install.rsp`) on the installation CD: The sample response file is in the `response_files` directory.
2. Copy the response file to the application server where you intend to install Telecom Web Services Server.
3. Open the Telecom Web Services Server base installer response file (`twss_base_install.rsp`) in a text editor of your choice.
4. Edit the response file properties.

   For the silent install to complete successfully, you must accept the license agreement by setting the parameter **-G LICENSE_ACCEPTED** to `true`. The following table lists the response file parameters that have to be configured to initially install all of the Telecom Web Services Server base components to the same target directory.

*Table 25. License acceptance panel*

| Property | Description | Example values |
|---|---|---|
| LICENSE_ACCEPTED | The license must be accepted before the installation process.<br>**Note:** This is not required for the uninstaller response file. | `true` |

*Table 26. Wizard input panel*

| Property | Description | Example values |
|---|---|---|
| WAS_HOME | The target installation home directory of the WebSphere server.<br>**Note:** This is not required for the uninstaller response file. | `/opt/IBM/WebSphere/AppServer` |

*Table 27. Telecom Web Services Server base installer features*

| Property | Description | Example values |
|---|---|---|
| spm | The selection state of the Service Policy Manager feature.<br>**Note:** If you select this feature, all of its sub-features will also be selected. | `true` or `false` |

*Table 27. Telecom Web Services Server base installer features (continued)*

| Property | Description | Example values |
|---|---|---|
| spm_runtime | The selection state of the Service Policy Manager runtime feature. This is the sub-feature of `spm`. | `true` or `false` |
| spm_console | The selection state of the Service Policy Manager console feature. This is the sub-feature of `spm`. | `true` or `false` |
| ag | The selection state of the Access Gateway feature.<br>**Note:** If you select this feature, all of its sub-features will also be selected. | `true` or `false` |
| ag_runtime | The selection state of the Access Gateway runtime feature. This is the sub-feature of `ag`. | `true` or `false` |
| ag_apps | The selection state of the Access Gateway applications feature. This is the sub-feature of `ag`.<br>**Note:** If you select this feature, all of its sub-features will also be selected. | `true` or `false` |
| ag_apps_ac_flows | The selection state of the Audio Calls feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_alm_flows | The selection state of the Address Llist Management flows feature. This is the sub-feature of`ag` and `ag_apps`. | `true` or `false` |
| ag_apps_am_flows | The selection state of the Account Management flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_ch_flows | The selection state of the Call Handling flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_cn_flows | The selection state of the Call Notification flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_mc_flows | The selection state of the Multimedia Conferencing flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |

*Table 27. Telecom Web Services Server base installer features (continued)*

| Property | Description | Example values |
|---|---|---|
| ag_apps_mm_flows | The selection state of the Multimedia Messaging flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_pmt_acs_flows | The selection state of the Payment (ACS) flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_pmt_rvcs_flows | The selection state of the Payment (RVCS) flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_prs_flows | The selection state of the Presence flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_sms_flows | The selection state of the SMS flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_tl_flows | The selection state of the Terminal Location flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_ts_flows | The selection state of the Terminal Status flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_tpc_flows | The selection state of the Third Party Call flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| ag_apps_wapp_flows | The selection state of the WAP Push call flows feature. This is the sub-feature of `ag` and `ag_apps`. | `true` or `false` |
| sp | The selection state of the Service Platform components feature.<br>**Note:** If you select this feature, then all of its sub-features will also be selected. | `true` or `false` |
| sp_runtime | The selection state of the Service Platform componentsruntime feature. This is the sub-feature of `sp`. | `true` or `false` |

*Table 27. Telecom Web Services Server base installer features (continued)*

| Property | Description | Example values |
|---|---|---|
| sp_pc | The selection state of the Parlay Connector feature. This is the sub-feature of `sp`. | `true` or `false` |
| sp_ac | The selection state of the Service Platform components Console feature. This is the sub-feature of `sp`. | `true` or `false` |
| sp_pac | The selection state of the Service Platform components feature in the Parlay Administration Console. This is the sub-feature of `sp`. | `true` or `false` |
| sp_cc | The selection state of the Service Platform components applications feature. This is the sub-feature of `sp`. | `true` or `false` |
| sp_wsdl | The selection state of the Service Platform components WSDLs feature This is the sub-feature of `sp`. | `true` or `false` |
| firstSteps | The selection state of the First Steps script. | `true` or `false` |

5. Save the edited document.

### Results

**Note:** The Telecom Web Services Server base uninstaller can also be run in silent mode. Copy and modify the parameters in the sample uninstall response file, `twss_base_uninstaller.rsp` from the CD, to indicate which features and components should be uninstalled.

### What to do next

You are now ready to run the installation script. The script will use the edited response file to get the settings it needs.

## Installing Telecom Web Services Server base components silently

Silent installations can be run by executing the setup script using a customized response file.

### Before you begin

Ensure that you have met the installation prerequisites that are listed in the topic *Customizing the installation response file for Telecom Web Services Server base components*. Then customize the response file using the steps that are found in the same topic.

### About this task

After customizing the response file, follow these steps to install the TWSS base components silently:

1. Log on to the operating system with a user ID that has administrator authority, such as **root**.
2. Select a umask that allows the owner to read and write to the files, and that allows others to access them according to the prevailing system policy.

   **Note:** For root, a umask of **022** is recommended. For non-root users an umask of **002** or **022** can be used, depending on whether or not the users share the group.
3. Invoke the setup script to install Telecom Web Services Server base components. The script makes use of your customized response file. Issue the following command:

   ```
   ./setup -i silent -f full_path_to_response_file
   ```

   For example:

   ```
   ./setup -i silent -f /opt/IBM/twss_base_install.rsp
   ```

   The installation process always creates an install trace file in the /tmp directory `TWSS-base_InstallTrace.log`. This file is helpful in determining any install problems and is required by the product support team for any install-related problems. After the installation is done, this trace file is moved to the *was_root*`/logs` directory.

   A separate log file is also created under the *was_root*`/logs` directory: `TWSS-base_InstallLog.log`. The log file is translated to user selected language during install.

### Results

Depending on the options selected, the silent installation may take a while to complete. For example, it may take time for the WebSphere Application Server to be stopped.

### What to do next

When the silent installation has completed, check the appropriate logs in the *was_root*`/logs` directory. Refer to the topic *Monitoring log messages* for additional information about reading logs.

## Verifying the results after installing Telecom Web Services Server base components

When the installation completes, you can check the installer log and check the installation directory to verify that the installation completed successfully.

### About this task

After you install the Telecom Web Services Server (TWSS) base components using the interactive installer, a message display on the summary panels to indicate that the installation was completed successfully. If you performed a silent installation, you can verify the installation results by viewing the installation log file at *was_root*`/logs/TWSS_Base-InstallLog.log`.

If the installation is unsuccessful, you can attempt to locate the reason by viewing the errors in the same log file. In addition to the log file, a trace file with more exclusive details can be found at this location: `/tmp/TWSS_Base-InstallTrace.log`

Both log and trace files are required when contacting IBM Support for any installation related issues.

In addition to log files, you should also verify the following for successful installation.

Navigate to the WebSphere home directory (*`was_root`*). You will see the following subdirectories:

- `Uninstall_TWSS-Base`
- `installableApps/TWSS-Base`

---

# Installing Telecom Web Services Server services

Use the installation script to install the Web service implementations.

You can either run the interactive installer, or you can perform a silent install.

## Installing Telecom Web Services Server services using the interactive installer

The Telecom Web Services Server (TWSS) service installer guides you through a set of interactive dialogs to install the Web service implementations.

### Before you begin

Ensure that the following requirements are met:

- You have installed WebSphere Application Server Network Deployment (version 7.0.0.1 or 6.1.0.21) with a deployment manager and at least one managed node.
- You have federated all nodes into the deployment manager cell.
- You have created a WebSphere user account repository. It is recommended that an LDAP repository be federated with the TWSS base installer.

  **Note:** See the WebSphere Application Server information center for details about performing these tasks.
- You have performed the database preparation steps described in the topic *Preparing for the Telecom Web Services Server database*.
- You have created clusters for the Telecom Web Services Server components you plan to install. Refer to the topic *Creating the clusters* for more information.
- You have completed the installation of TWSS base components. (See the *Installing Telecom Web Services Server base components* section of this information center.)

The TWSS services installer installs all of the available Web service implementations. It moves database files to the features with which they are associated, rather than to a standalone database server.

1. Navigate to the `/TWSS_Install` directory on the CD.
2. Launch the services installation wizard by issuing the following command:
   ```
   ./setup
   ```
   The installation process always creates an install trace file in the /tmp directory `TWSS-Services_InstallTrace.log`. This file is helpful in determining any install problems and is required by the product support team for any install-related problems. After the installation is done, this trace file is moved to the *`was_root`*`/logs` directory.

   A separate log file is also created under the *`was_root`*`/logs` directory: `TWSS-Services_InstallLog.log`. The log file is translated to user selected language during install.

3. The Welcome panel displays with English as the default. Click **OK**.
4. Read the license agreement. Then select **I accept the terms in the license agreement**, and click **Next**.
5. Review the information in the Product information panel and click **Next**.
6. Specify the WebSphere Application Server home directory. This must be your `was_root` directory.
7. Click **Next**.
8. In the Features panel, select the components you want to install, and click **Next**.
9. In the Summary Information panel, verify your selections. Use the **Back** button to go back and make changes. When you are satisfied with your selections, click **Next**.
10. When the installation is complete, click **Finish** to exit the wizard.

### What to do next

To check that the installation was successful, refer to the topic *Verifying the results after installing Telecom Web Services Server services*.

## Installing Telecom Web Services Server Web services silently

The Telecom Web Services Server (TWSS) Web service implementations can be installed silently without user interaction, by reading user responses from a response file.

### About this task

A silent installation uses the TWSS services installer to install the product in silent mode, without the graphical user interface. Instead of displaying a wizard interface, the silent installation causes the installation program to read all of your responses from a response file that you provide.

To specify non-default options during a silent installation, customize the response file. To install silently, you must accept the license agreement using the agreement option within the response file.

### Customizing the installation response file for Telecom Web Services Server Web service implementations

Before invoking the Telecom Web Services Server (TWSS) services installer in silent mode, you need to customize the response file that will be used to provide user responses to the installer.

#### Before you begin

Ensure that the following requirements are met:

- You have installed WebSphere Application Server Network Deployment (version 7.0.0.1 or 6.1.0.21) with a deployment manager and at least one managed node.
- You have federated all nodes into the deployment manager cell.
- You have created a WebSphere user account repository. It is recommended that an LDAP repository be federated with the TWSS base installer.

  **Note:** See the WebSphere Application Server information center for details about performing these tasks.

- You have performed the database preparation steps described in the topic *Preparing for the Telecom Web Services Server database*.
- You have created clusters for the Telecom Web Services Server components you plan to install. Refer to the topic *Creating the clusters* for more information.
- You have completed the installation of TWSS base components. (See the *Installing Telecom Web Services Server base components* section of this information center.)

## About this task

Use the response file to supply values to the TWSS services installer. Running in silent mode, the installer does not display interactive dialogs. Instead, it reads values from the response file.

To modify the install response file, you must copy sample response file, `twss_services_install.rsp`, from the CD.

**Note:** If the installer fails before or while processing the parameter that specifies the target directory, the log file will be in the system's temporary directory.

To customize the response file in preparation for a silent installation of the TWSS Web service implementations, follow these steps.

1. Locate the response file (`twss_services_install.rsp`) on the installation CD: The sample response file is in the `response_files` directory.
2. Copy the response file to the application server where you intend to install Telecom Web Services Server.
3. Open the Telecom Web Services Server services installer response file (`twss_services_install.rsp`) in a text editor of your choice.
4. Edit the response file properties.

   For the silent install to complete successfully, you must accept the license agreement by setting the parameter **-G LICENSE_ACCEPTED** to `true`. The following table lists the response file parameters that have to be configured to initially install all of the Telecom Web Services Server Web services to the same target directory.

*Table 28. License acceptance panel*

| Property | Description | Example values |
|---|---|---|
| LICENSE_ACCEPTED | The license must be accepted before the installation process.<br>**Note:** This is not required for the uninstaller response file. | `true` |

*Table 29. Telecom Web Services Server services installer - Wizard input panels*

| Property | Description | Example values |
|---|---|---|
| WAS_HOME | The home target installation directory of a supported WebSphere server.<br>**Note:** This is not required for the uninstaller response file. | `/opt/IBM/WebSphere/AppServer` |

*Table 30. Telecom Web Services Server services installer - features*

| Property | Description | Example values |
|---|---|---|
| ims_cn | The selection state of the Call Notification over SIP/IMS feature. | `true` or `false` |
| ims_pres | The selection state of the Presence over SIP/IMS feature. | `true` or `false` |
| ims_ts | The selection state of the Terminal Status over SIP/IMS feature. | `true` or `false` |
| pay | The selection state of the Payment (Postpaid) over Usage Records/CEI feature. | `true` or `false` |
| xcap_alm | The selection state of the Address List Management over XCAP feature. | `true` or `false` |
| smpp_sms | The selection state of the Short Message Service over SMPP feature. | `true` or `false` |
| smpp_wap | The selection state of the Wireless Access Protocol (WAP) Push over SMPP feature. | `true` or `false` |
| mlp_tl | The selection state of the Terminal Location over MLP feature. | `true` or `false` |
| mm7_mms | The selection state of the Multimedia Messaging Service over MM7 feature. | `true` or `false` |
| parlay_ch | The selection state of the Call Handling over Parlay feature. | `true` or `false` |
| parlay_cn | The selection state of the Call Direction/Notification over Parlay feature. | `true` or `false` |
| parlay_sms | The selection state of the Short Message Service over Parlay feature. | `true` or `false` |
| parlay_tl | The selection state of the Terminal Location over Parlay feature. | `true` or `false` |
| parlay_ts | The selection state of the Terminal Status over Parlay feature. | `true` or `false` |
| parlay_tpc | The selection state of the Third Party Call over Parlay feature. | `true` or `false` |

5. Save the edited document.

### Results

**Note:** The Telecom Web Services Server services uninstaller can also be run in silent mode. Copy and modify the parameters in the sample response file, `twss_services_uninstaller.rsp` from the CD, to indicate which features and components should be uninstalled.

### What to do next

You are now ready to run the installation script. The script will use the edited response file to get the settings it needs.

## Installing Telecom Web Services Server Web service implementations silently

Silent installations can be run by executing the setup script using a customized response file.

### Before you begin

Ensure that you have met the installation prerequisites that are listed in the topic *Customizing the installation response file for Telecom Web Services Server Web service implementations*. Then customize the response file using the steps that are found in the same topic.

### About this task

Follow these steps to install the TWSS Web service implementations silently.

1. Log on to the operating system with a user ID that has administrator authority, such as **root**.
2. Select a umask that allows the owner to read and write to the files, and that allows others to access them according to the prevailing system policy.

   **Note:** For root, a umask of **022** is recommended. For non-root users an umask of **002** or **022** can be used, depending on whether or not the users share the group.
3. Invoke the setup script to install Telecom Web Services Server services. The script makes use of your customized response file. Issue the following command:

   ```
   setup -silent -options full_path_to_response_file
   ```

   For example:

   ```
   ./setup -i silent -f /opt/IBM/twss_services_install.rsp
   ```

   The installation process always creates an install trace file in the /tmp directory `TWSS-base_InstallTrace.log`. This file is helpful in determining any install problems and is required by the product support team for any install-related problems. After the installation is done, this trace file is moved to the *was_root*`/logs` directory.

   A separate log file is also created under the *was_root*`/logs` directory: `TWSS-base_InstallLog.log`. The log file is translated to user selected language during install.

### What to do next

When the silent installation has completed, check the appropriate logs in the `was_root/logs` directory. Refer to the topic *Monitoring log messages* for additional information about reading logs.

## Verifying the results after installing Telecom Web Services Server services

When the installation completes, you can check the installer log and check the installation directory to verify that the installation completed successfully.

### About this task

After you install the Telecom Web Services Server (TWSS) services using the interactive installer, a message display on the summary panels to indicate that the installation was completed successfully. If you performed a silent installation, you can verify the installation results by viewing the installation log file at `was_root/logs/TWSS_Base-InstallLog.log`.

If the installation is unsuccessful, you can attempt to locate the reason by viewing the errors in the same log file. In addition to the log file, a trace file with more exclusive details can be found at this location: `/tmp/TWSS_Services-InstallTrace.log`

Both log and trace files are required when contacting IBM Support for any installation related issues.

In addition to log files, you should also verify the following for successful installation.

Navigate to the WebSphere home directory (*was_root*). You will see the following subdirectories:

- `Uninstall_TWSS-Services`
- `installableApps/TWSS-Services`

# Configuring your installed system

After you install the Telecom Web Services Server base components and services, some additional customization is needed to get your system up and running. The First Steps script performs most of the customization work.

You can run the First Steps script in interactive mode, in which you respond to a series of prompts before the configuration takes place, or you can configure a response file and run the wizard silently.

## Creating and configuring the DB2 database server instance

If you are using a DB2 database, you must create and configure the database server instance. After you perform this task, you will run the First Steps script to create the necessary database tables and complete your configuration.

### Before you begin

Before you begin, you must have completed the following tasks:

- Installed and started IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1

- Performed the database configuration steps in the topic *Preparing to set up the database on DB2*
- Installed Telecom Web Services Server base components on the database server

## About this task

Complete the following steps on the database server. If you are using a distributed database configuration, complete the steps on *each* server that will host a database.

1. Log in to the database server as **root**.
2. Copy the contents of the following applicable directories from the server where you installed the TWSS base components to the corresponding location on the database server.
   - `*was_root*/installableApps/TWSS-Base/database`, where `*was_root*` represents the WebSphere Application Server installation location (for a shared database)
   - *esb_root*`/installableApps/TWSS-Base/database`, where *esb_root* represents the installation location of the WebSphere ESB (for a distributed database, Access Gateway migration)
   - `*was_root*/installableApps/TWSS-Services/database` (for a distributed database, MMS/MM7, SMS/SMP and WAPPUSH/SMPP migration)

   **Note:** This step is not necessary if the database server is located on the same physical server as the TWSS base components.
3. Run the following command to change the permissions on the database configuration script.
   ```
   chmod 755 crtsrvDb2.sh
   ```
4. From the command line, switch to a user ID with database administrator authority, such as db2inst1. For example, type:
   ```
   su - db2inst1
   ```
5. If you are not already in the directory to which you copied the database setup files, change to it. Example:
   ```
   cd was_root/installableApps/TWSS-Base/database
   ```
6. Launch the crtsrvDb2 script using the following command:
   ```
   ./crtsrvDb2.sh options
   ```

   where: *options* corresponds to the following values.

*Table 31. `crtsrvDb2` configuration values*

| Parameter | Description | Recommended Value |
|---|---|---|
| dbServer | Fully qualified hostname for the database server. | Example: `myhost.example.com` |
| dbPort | Database server connection port. | 50000 |
| dbLocal | TRUE if the database server is located on the same physical server as the TWSS base components.<br>FALSE if the database server is on a different server. This is the typical configuration. | TRUE |
| dbNodeName | Remote database server node name, when dbLocal is FALSE. This value is used for reference purposes and it is not the host name. | RDBSRV |

*Table 31. `crtsrvDb2` configuration values (continued)*

| **Parameter** | **Description** | **Recommended Value** |
|---|---|---|
| dbName | Name of the database. If you are using a consolidated or shared database configuration, the recommended name is `TWSS70`. If you are using a distributed database configuration, use a name that describes the function of this database–for example, `AGDB` for the Access Gateway. | TWSS70 |
| dbAlias | Database alias. If you are using a consolidated or shared database configuration, the recommended name is `TWSS70A`. If you are using a distributed database configuration, use a name that describes the function of this database–for example, `AGDBA`. | TWSS70A |
| dbLocale | Database locale or territory code. | US |
| dbAdmin | Administrator user ID for the database instance. | Example: `db2inst1` |
| dbAdminPW | Password for the administrator user ID. | not applicable |
| dbUser | User ID for accessing the database. | twssuser |
| dbUserPW | Password for the database user ID. | not applicable |
| INITorDDLFile | One of the following:<br>INIT to initialize the database instance<br>The path to the DDL file | INIT |
| infoflag | TRUE to display informational messages while the database is being initialized.<br>FALSE to suppress informational messages. | TRUE |

Example (options are split across lines for ease in reading):

```
./crtsrvDb2.sh myhost.example.com 50000 TRUE RDBSRV TWSS70 TWSS70A
US db2inst1 password twssuser password INIT TRUE
```

### Results

When the crtsrvDb2 command completes, the database server instance has been configured.

### What to do next

Next, you will run the First Steps script to complete the configuration for Telecom Web Services Server. The First Steps script will create the necessary database tables.

## Creating and configuring the Oracle database server instance

If you are using an Oracle database, you must create and configure the database server instance. After you perform this task, you will run the First Steps script to create the necessary database tables and complete your configuration.

### Before you begin

Before you begin, you must have completed the following tasks:

- Installed and started a supported version of Oracle Database on the database server
- Performed the database configuration steps in the topic *Preparing to set up the database on Oracle*

- Installed Telecom Web Services Server on the database server

## About this task

Complete the following steps on the database server. If you are using a distributed database configuration, complete the steps on *each* server that will host a database.

1. Log in to the database server as **root**.
2. Copy the contents of the following applicable directories from the server where you installed the TWSS base components to the corresponding location on the database server.
   - *`was_root`*`/installableApps/TWSS-Base/database`, where *`was_root`* represents the WebSphere Application Server installation location (for a shared database)
   - *esb_root*`/installableApps/TWSS-Base/database`, where *esb_root* represents the installation location of the WebSphere ESB (for a distributed database, Access Gateway migration)
   - *`was_root`*`/installableApps/TWSS-Services/database` (for a distributed database, MMS/MM7, SMS/SMP and WAPPUSH/SMPP migration)

   **Note:** This step is not necessary if the database server is located on the same physical server as the TWSS base components.
3. Run the following command to change the permissions on the script.
   ```
   chmod 755 crtsrvOra.sh
   ```
4. Open a command prompt and switch to the Oracle database administrator ID.
   ```
   su - oracle
   ```

   **Note:** This ID is `oracle` by default. If you use a different ID, modify the command accordingly.
5. If you are not already in the directory to which you copied the database setup files, change to it. Example:
   ```
   cd was_root/installableApps/TWSS-Base/database
   ```
6. Launch the `crtsrvOra` script by issuing the following command:
   ```
   ./crtsrvOra.sh options
   ```

   where: *options* corresponds to the following values.

*Table 32. `crtsrvOra` configuration values*

| Parameter | Description | Recommended Value |
|---|---|---|
| dbName | Database name. If you are using a consolidated or shared database configuration, the recommended name is `TWSS70`. If you are using a distributed database configuration, use a name that describes the function of this database–for example, `AGDBD` for the Access Gateway. | TWSS70 |
| dbAdmin | Administrator user ID for the database instance, for example `system`. | not applicable |
| dbAdminPW | Password for the administrator user ID. | not applicable |
| dbUser | User ID for accessing the database. | twssuser |
| dbUserPW | Password for the database user ID. | not applicable |
| INIT | Use INIT to initialize the database instance. | INIT |

*Table 32. `crtsrvOra` configuration values (continued)*

| Parameter | Description | Recommended Value |
|---|---|---|
| infoflag | TRUE to display informational messages while the database is being initialized.<br>FALSE to suppress informational messages. | TRUE |

Example (options are split across lines for ease in reading):

```
./crtsrvOra.sh TWSS70 system admin_password twssuser
user_password INIT TRUE
```

### Results

When the crtsrvOra command completes, the database server instance has been configured.

### What to do next

Next, you will run the First Steps script to complete the configuration for Telecom Web Services Server. The First Steps script will create the necessary database tables.

# Running the First Steps configuration script

The First Steps script performs basic, default configuration steps for the Telecom Web Services Server (TWSS) components you have installed.

The First Steps script creates tables for a single, shared database that is shared by all of the TWSS components. If you prefer to use a distributed database configuration, you will need to configure the databases manually as described in the topic *Migrating data in a distributed database configuration*.

It is important to note that the First Steps script deploys and configures all of the base components and services that are installed on the server where it runs. Configurations in which a single server is shared between multiple clusters, and a different set of services is deployed on each cluster, are not supported.

**Note:** Use of the First Steps script with the IBM HTTP Server (IHS) and the HTTP Web server plug-in is not supported. As an alternative, use the WebSphere Proxy Server.

## Running the First Steps script interactively

You can run the First Steps script in interactive mode, responding to prompts as the configuration proceeds.

### Before you begin

- You must have configured the DB2 or Oracle database instance as described in the topic *Creating and configuring the database server instance* for your database type.
- You must have installed the Telecom Web Services Server (TWSS) base components and Web service implementations.
- The CLASSPATH environment variable must contain the Java libraries for DB2 or Oracle.
- The PATH environment variable must contain the binaries for DB2 or Oracle.

## About this task

The configuration properties file is found at the following directory location: *was_root*`/installableApps/TWSS-Base/firststeps/firststeps.properties`. The First Steps script launches a wizard that updates this properties file with the values you specify.

**Note:** If you are migrating from an earlier release of TWSS, the First Steps script will migrate data only from a combined database, not from distributed databases. For information about migrating data when you use a distributed database configuration, see the topic *Migrating data in a distributed database configuration*.

Follow these steps to perform basic configuration for your newly installed TWSS components:

1. Start the deployment manager and all associated node agents.
2. On the deployment manager where the TWSS base components have been installed, log in using `root` or another valid ID.
3. Navigate to the following directory: *was_root*`/installableApps/TWSS-Base/firststeps`.
4. If the database is installed on the deployment manager server, issue an export command similar to one of the following examples to ensure that the Java classpath environment variable includes appropriate JDBC driver JAR files for your database.

   DB2 `export db2=/opt/ibm/db2/V9.5`

   DB2 `export CLASSPATH=${CLASSPATH}:${db2}/java/db2java.zip:${db2}/java/db2jcc.jar:${db2}/java/db2jcc_license_cu.jar`

   Oracle `export CLASSPATH=${CLASSPATH}:/Oracle/opt/oracle/product/11.1.0/client_1/jdbc/lib`
5. If the database is installed on a separate server, follow these steps:
   a. On the deployment manager and on each TWSS node, create a new directory to hold the database driver files, for example `/db2_driver` or `/oracle_driver`.

   b. DB2 FTP or copy the following files from the DB2 server to each of the new directories you created:

      `db2jcc_license_cu.jar`

      `db2jcc.jar`

      `db2java.zip`

   c. DB2 Issue an export command similar to the following:

   ```
   export CLASSPATH=${CLASSPATH}:/db2_driver/db2java.zip:/db2_driver/db2jcc.jar:/db2_driver/
   ```

   d. Oracle FTP or copy the file `ojdbc5.jar` from the Oracle server to each of the new directories you created.

   e. Oracle Issue an export command similar to the following:

   ```
   export CLASSPATH=${CLASSPATH}:/oracle_drivers/ojdbc5.jar
   ```
6. Configure a target scope for the Access Gateway, the Service Platform, and the Service Policy Manager.

Each of these components can be configured on a WebSphere topology that is managed by the deployment manager on which you are running the First Steps script.

- To configure a component on a cluster, type the cluster name in the appropriate target scope.
- To configure a component on a server, either standalone or managed by the deployment manager, use the slash character–for example `node/server`.
- To bypass configuration for the component altogether, type `None`.

For example, by running the First Steps script just once, you can configure the Access Gateway on `TWSSCluster_1` and the other two components on `TWSSCluster_2`, provided both clusters are managed by the same deployment manager. However, if the clusters are managed by two separate deployment managers, you must run the First Steps script twice–once on each deployment manager.

7. Issue the following command to launch the First Steps script:

```
./firststeps.sh @firststeps.properties wsadmin_options
```

where *wsadmin_options* are optional parameters that will be passed to the wsadmin command, which is invoked by the script. For example, if WebSphere global security is enabled, you can supply the user name and password:`-username wasadmin -password was123`.

8. In the **Configuration Mode** section, select the appropriate option:

**Initial Configuration Mode**

For a first-time or clean installation, initial values are provided for all configuration resources and properties. Existing values, if any, are overwritten.

**Note:** This option should only be used during the first-time deployment of any of the Telecom Web Services Server components. If there are subsequent add-ons of any of the new components, then the update option should be used. This option completely drops all the tables in the database and recreates.

**Migrate**

For upgrading from a previous version of the product, initial values are provided for the missing configuration information. Existing configuration resources and properties are retained whenever possible.

**Note:** This option supports only the consolidated database setup of Telecom Web Services Server 6.2 release.

**Update**

This option should be used when deploying service add-ons to the existing setup. It also applies when making updates to an interim fix (iFix) or fix pack if you are so instructed by the Readme. Initial values are provided for missing configuration resources and properties, while existing configuration resources and properties are retained whenever possible. This option does not drop existing tables but creates the newly required tables in the database.

**Note:** The existing database is retained only if all of the following values you supply in step 9 on page 165 match the values for the existing database: server type, server host name, server port, and database name. If any of these values do not match, the First Steps script replaces the existing database.

**Note:** If the Service Platform components are installed using the *Initial* option and you would like to add, for example, the Service Policy Manager or the Access Gateway using the same database, then you should use the *Update* option for the Service Policy Manager or the Access Gateway.

9. In the **Database Configuration** section, complete the following fields:

**Database Server Type**
Either `DB2` or `Oracle`. The default value is `DB2`.

**Database Server Access Mode**
One of the following:

**Local** The database server is on the same machine as the deployment manager, and the only server instance is also on the same machine.

**Remote**
The database server is on another machine, and all nodes will access it remotely using the hostname. This is the default value.

**Note:** The database NodeName for DB2 will be set to a constant `RDBSRV` for a remote database server, or `local` for a local database.

**Database Server Hostname**
The host name or IP address for the database server. The default value is `localhost`.

**Database Server Port**
The port number through which the application server will access the database server. The default value is `50000` for a DB2 server, and `1521` for an Oracle server.

**Database Driver Path**
The value you want to use for the two WebSphere environment variables.

These examples assumes that the database is installed on the deployment manager server:

DB2 `/opt/ibm/db2/V9.5`

Oracle `/opt/oracle/product/11.1.0/client_1/jdbc/lib`

**Note:** The path is assumed to be the same for each node in a deployment. If different paths are used for different nodes, you will need to configure the WebSphere variables manually.

**Database Administrator Name**
The name of the Administrator user on the database server. This user name is used when initializing the database instance.

Examples:

DB2 `db2admin` or `db2inst1`

Oracle `SYSTEM`

**Database Administrator Password**
Password for the Administrator user on the database server.

**Database User Name**
The name of the user that will access the database server during runtime, for example `twssuser`.

**Database User Password**
Password for the user that will access the database server during runtime.

**Database Name**
The name of the database, no more than 6 characters in length. The default value is `TWSS70`.

This value will be appended with `D` to generate the full database name, and appended with `A` to generate the full alias name. The full name is limited by some databases to 7 or 8 characters.

In a distributed database configuration, use the name of the database that applies to this cluster – for example, AGDB for the Access Gateway cluster or SPMDB for the Service Policy Manager cluster.

**Database Locale**
The locale string used to initialize the database. The default value is `US`.

10. In the **WAS Configuration** section, complete the following fields to provide information about your WebSphere setup:

**WAS Administrator User Name**
The administrative user name for WebSphere Application Server. Required if WebSphere security is enabled.

**WAS Administrator Password**
The password for the administrative user. Required if WebSphere security is enabled.

**Access Gateway Target Scope**
The target scope for the Access Gateway: either the name of a cluster, a name in the form *`node/server`*, or `None`.

**Service Platform Target Scope**
The target scope for the Service Policy Manager: either the name of a cluster, a name in the form *`node/server`*, or `None`.

**Service Policy Manager Target Scope**
The target scope for the Service Platform components: either the name of a cluster, a name in the form *`node/server`*, or `None`.

**WC Default Host Port**
The port used by the Web container for Web service implementations. Use this port number in all URLs referenced in the configuration.

The default value is `9080`. However, you may need to specify `9081` or `9082` if there are other servers on the same physical machine.

**JMS Queue Mode**
If a file store or data store is used, the root path where the bus members store their data. The default value is **filestore**.

**JMS Queue Path**
The full path to a directory where file store JMS queues can be stored. In a clustered configuration, this directory must be on an NFS

mounted file system so that it can be shared among all JMS queue engines. For a single node configuration, it can be a local directory. There is no default value.

**JMS Datastore JNDI Name**
The JNDI name of a JDBC data source where the data store's JMS queue data is stored. The default value is `jdbc\TWSSDB`.

**JMX Administrator User Name**
A user ID with WebSphere Application Server Administrator authority. This name is used by the Access Gateway and the Service Platform components to emit JMX notifications. Required if WebSphere security is enabled.

**JMX Administrator Password**
Password for the WebSphere Application Server Administrator user.

11. In the **TWSS Configuration** section, complete the following fields to provide information about your Telecom Web Services Server (TWSS) setup:

**TWSS User Name**
A user ID used to access the Service Platform components and the Access Gateway from the PX Notification component Web service. Required if basic authentication is used to authenticate between the Service Platform components and the Access Gateway.

**TWSS User Password**
Password for the TWSS user.

**SPM Policy Access URL**
The endpoint URL for the Service Policy Manager Policy Access interface. The default value is http: `//localhost:9080/SPM/Access/services/PolicyAccess`.

**Group Resolution URL**
The endpoint URL for the Address List Management interface. The default value is `http://localhost:9080/TWSS/ParlayX21/Admin/AddressListManagement/IMS/services/AddressListManagement`.

**Privacy Implementation Enabled**
`true` if you are installing a privacy implementation, `false` if not. The default value is `true`.

**Access Gateway Root URL**
The service endpoint root for the Access Gateway, in the form *hostname*:*port*. This value is common to all services, and it will also be the URL for the load balancer if desired. The default value is `http://localhost:9080`.

**Service Platform Root URL**
The service endpoint root for the Service Platform components, in the form *hostname*:*port*. This value is common to all services. The default value is `http://localhost:9080`.

**Policy Administrator Group Name**
The group name from the user registry used for the PolicyAdministrator role.

**TWSS Administrator Group Name**
The group name from the user registry used for the TWSS Administration Console.

**Notify Mgmt Administrator Group Name**
The group name from the user registry used for the Notification Management Administration interface.

12. Optional: If you want to save your values into a configuration properties file, to be used for a silent installation at a later time, click **Save**.
13. To start the configuration process, click **Configure**.

### Results

The First Steps script performs all of the following configuration steps:

- Initializes the necessary databases.
- Configures the common WebSphere Application Server resources.
- Configures the common Access Gateway resources.
- Configures the common Service Platform resources.
- Configures the resources for each installed Service Platform component and Web service implementation.
- Performs the same configuration for all server instances associated with the cluster.
- Configures the server and the node specific configuration items.

### What to do next

When the script completes, it might be necessary to perform some additional configuration, depending on which Web service implementations you plan to deploy. For more information, refer to the topic *Configuring connections for the Web service implementations*.

To check the results of running the First Steps script, refer to the topic *Verifying the completed configuration*.

## Performing the configuration silently

For subsequent installations after you have run the First Steps script, or simply as an alternative to using the interactive wizard, you can configure silently.

### Before you begin

- You must have configured the DB2 or Oracle database instance as described in the topic *Creating and configuring the database server instance* for your database type.
- You must have installed the Telecom Web Services Server (TWSS) base components and Web service implementations.
- The CLASSPATH environment variable must contain the Java libraries for DB2 or Oracle.
- The PATH environment variable must contain the binaries for DB2 or Oracle.

### About this task

The configuration properties file is found at the following directory location: *`was_root`*`/installableApps/TWSS-Base/firststeps/firststeps.properties`.

**Note:** If you are migrating from an earlier release of TWSS, the First Steps script will migrate data only from a combined database, not from distributed databases. For information about migrating data when you use a distributed database configuration, see the topic *Migrating data in a distributed database configuration*.

1. Start the deployment manager and all associated node agents.
2. On the deployment manager server where the Telecom Web Services Server base components have been installed, log in with the same WebSphere Administrator ID that was used for the installation.
3. Navigate to the following directory: `*was_root*/installableApps/TWSS-Base/firststeps`.
4. Open the `firststeps.properties` file in a text editor, then edit the configuration properties. The following table lists a representative sample of the configuration file parameters. The detailed descriptions for all of the parameters can be found in the configuration file itself.

   **Note:** Do not populate the password fields. You will supply all of the passwords as command line parameters at runtime.

*Table 33. Configuration file properties*

| Property | Type | Description |
|---|---|---|
| PropertyOrder | hidden | A comma-delimited list (name1, name2, name3) of key values that define the presentation order of the properties in the Configuration Information Window. The default order is the order listed in this document. |
| Firststeps Configuration | NamedSeparator | Separator bar with the value of `Firststeps Configuration`. |

*Table 33. Configuration file properties (continued)*

| **Property** | **Type** | **Description** |
|---|---|---|
| ConfigurationMode | choice | One of the following:<br>**Initial** First-time installation of Telecom Web Services Server. All configuration values are reset according to the data you specify. Associated policies, configuration data tables, and runtime state tables are reset.<br>**Migrate** Migration from a previous version. Existing configuration values are used whenever possible; new values are used when no existing values are found. Associated policies and configuration data tables are preserved, but runtime state tables are reset.<br>**Update** Installation of a service update, such as an interim fix or fix pack. Use when instructed to do so in a readme file. Existing configuration values are used whenever possible; new values are used when no existing values are found. Associated policies, configuration data tables, and runtime state tables are preserved.<br>**Note:** The existing database is retained only if all of the following values in the response file match the values for the existing database: server type, server host name, server port, and database name. If any of these values do not match, the First Steps script replaces the existing database. |
| DatabaseConfiguration | named separator | A separator bar with the value of `Database Configuration`. |
| DatabaseServerType | choice | Either `DB2` or `Oracle`. The default value is `DB2`. |

*Table 33. Configuration file properties (continued)*

| Property | Type | Description |
|---|---|---|
| DatabaseServerAccessMode | choice | One of the following:<br>**Local** The database server is on the same machine as the deployment manager, and the only server instance is also on the same machine.<br>**Remote** The database server is on another machine, and all nodes will access it remotely using the hostname. This is the default value.<br>**Note:** The database NodeName for DB2 will be set to a constant `RDBSRV` for a remote database server, or `local` for a local database. |
| DatabaseServerHostname | string | The host name or IP address for the database server. The default value is `localhost`. |
| DatabaseServerPort | integer | The port number through which the application server will access the database server. The default value is `50000` for a DB2 server, and `1521` for an Oracle server. |
| DatabaseDriverRootPath | directory | The full path to the directory that contains JDBC driver JAR files for the database client. Specify the same value as the WebSphere environment variable DB2UNIVERSAL_JDBC_DRIVER_PATH or ORACLE_JDBC_DRIVER_PATH.<br>Examples:<br>DB2 `/opt/ibm/db2/V9.5`<br>Oracle `/opt/oracle/product/11.1.0/client_1/jdbc/lib`<br>**Note:** The path is assumed to be the same for each node in a deployment. If different paths are used for different nodes, you will need to configure the WebSphere variables manually. |
| DatabaseServerAdministratorUserName | string | The name of the Administrator user on the database server. This user name is used when initializing the database instance.<br>Examples:<br>DB2 `db2admin` or `db2inst1`<br>Oracle `SYSTEM` |
| DatabaseServerAdministratorPassword | password | Do not populate this field. |
| DatabaseServerUserName | string | The name of the user that will access the database server during runtime, for example `twssuser`. |
| DatabaseServerUserPassword | password | Do not populate this field. |

*Table 33. Configuration file properties (continued)*

| Property | Type | Description |
|---|---|---|
| DatabaseName | string | The name of the database, no more than 6 characters in length. The default value is `TWSS70`.<br>This value will be appended with `D` to generate the full database name, and appended with `A` to generate the full alias name. The full name is limited by some databases to 7 or 8 characters.<br>In a distributed database configuration, use the name of the database that applies to this cluster – for example, AGDB for the Access Gateway cluster or SPMDB for the Service Policy Manager cluster. |
| DatabaseLocale | string | The locale string used to initialize the database. The default value is `US`. |
| WASConfiguration | named separator | A separator bar with the value of `WAS Configuration`. |
| WASAdminUserName | string | The Administrator user name for WebSphere Application Server. |
| WASAdminPassword | password | Do not populate this field. |
| AGTargetScope | string | The target scope for the Access Gateway: either the name of a cluster, a name in the form *`node/server`*, or `None`. |
| SPMTargetScope | string | The target scope for the Service Policy Manager: either the name of a cluster, a name in the form *`node/server`*, or `None`. |
| SPTargetScope | string | The target scope for the Service Platform components: either the name of a cluster, a name in the form *`node/server`*, or `None`. |
| WCDefaultHostPort | integer | The port used by the Web container for Web service implementations. Use this port number in all URLs referenced in the configuration.<br>The default value is `9080`. However, you may need to specify `9081` or `9082` if there are other servers on the same physical machine. |

*Table 33. Configuration file properties (continued)*

| Property | Type | Description |
|---|---|---|
| CommonRuntimeFiles | hidden | A comma-separated list of relative paths to runtime files that are copied to the deployment manager, used for publishing the files to all server instances. The paths are relative to `was_root`.<br>Example:<br>plugins/ com.ibm.soa.common.reservation_1.0.0.jar,<br>lib/ext/common-statscache.jar<br>lib/ext/spm-ws-access-client.jar,<br>lib/ext/spm-ws-admin-client.jar,<br>properties/version/TWSS.product,<br>properties/version/TWSS.sp.component,<br>properties/version/TWSS.esb.component,<br>properties/version/TWSS.services.component,<br>properties/version/nif/componentmaps/componentmap.twss |
| AGRuntimeFiles | hidden | A comma-separated list of relative paths to Access Gateway runtime files that are copied to the deployment manager, used for publishing the files to all server instances. The paths are relative to `was_root`.<br>Example:<br>lib/ext/addressmasking.jar,<br>lib/ext/common-utils.jar,<br>lib/ext/esb-common-utils.jar,<br>lib/ext/esb-handlers.jar,<br>lib/ext/jmxnotification.jar,<br>lib/ext/msgelementremover.jar,<br>lib/ext/groupresolution.jar,<br>lib/ext/networkstatistics.jar,<br>lib/ext/policyretrieval.jar,<br>lib/ext/serviceauthorization.jar,<br>lib/ext/serviceinvocation.jar |
| JMSQueuePath | directory | The full path to a directory where file store JMS queues can be stored. In a clustered configuration, this directory must be on an NFS mounted file system so that it can be shared among all JMS queue engines. For a single node configuration, it can be a local directory. There is no default value. |
| JMSQueueMode | choice | If a file store or data store is used, the root path where the bus members store their data. The default value is `file store`. |
| JMSDatastoreJNDIName | string | The JNDI name of a JDBC data source where the data store's JMS queue data is stored. The default value is `jdbc\TWSSDB`. |
| JMXAdminUserName | string | A user ID with WebSphere Application Server Administrator authority. This name is used by the Access Gateway and the Service Platform components to emit JMX notifications. Required if WebSphere security is enabled. |

*Table 33. Configuration file properties (continued)*

| **Property** | **Type** | **Description** |
|---|---|---|
| JMXAdminPassword | password | Do not populate this field. |
| TWSSConfiguration | named separator | A separator bar with the value of `TWSS Configuration`. |
| TWSSUserName | string | A user ID used to access the Service Platform components and the Access Gateway from the PX Notification component Web service. Required if basic authentication is used to authenticate between the Service Platform components and the Access Gateway. |
| TWSSUserPassword | password | Do not populate this field. |
| SPMPolicyAccessURL | URL | The endpoint URL for the Service Policy Manager Policy Access interface. The default value is `http://`*`hostname`*`:`*`port`*`/SPM/Access/services/PolicyAccess`. |
| GroupResolutionURL | URL | The endpoint URL for the Address List Management interface. The default value is `http://`*`hostname`*`:`*`port`*`/TWSS/ParlayX21/Admin/AddressListManagement/IMS/services/AddressListManagement`. |
| PrivacyImplementationEnabled | Boolean | `true` if you are installing a privacy implementation; `false` if not. |
| AccessGatewayRootURL | URL | The service endpoint root for the Access Gateway, in the form *hostname*:*port*. This value is common to all services, and it will also be the URL for the load balancer if desired. The default value is `http://localhost:9080`. |
| ServicePlatformRootURL | URL | The service endpoint root for the Service Platform components, in the form *hostname*:*port*. This value is common to all services. The default value is `http://localhost:9080`. |
| PolicyAdminGroupName | string | The group name from the user registry used for the PolicyAdministrator role. |
| TWSSAdminGroupName | string | The group name from the user registry used for the TWSS Administration Console. |
| NotifyAdminGroupName | string | The group name from the user registry used for the Notification Management Administration interface. |

5. Save your changes and close the text editor.
6. Perform the configuration by launching the First Steps script. Enter the command parameters on a single line:

   ```
   ./firststeps.sh @firststeps.properties -silent -DBServerAdminPass db_admin_pw -DBServerUserPass d
   -WASAdminPass was_admin_pw -JMXAdminPass jmx_admin_pw -TWSSUserPass twss_user_pw wsadmin_options
   ```

   Where:

   > `firststeps.properties` is the name of your saved configuration file.
   >
   > *db_admin_pw* is the password for the Administrator user on the database server.

*db_user_pw* is the password for the user that will access the database server during runtime, for example `twssuser`.

*was_admin_pw* is the password for the WebSphere Application Server Administrator user.

*jmx_admin_pw* is the password for the WebSphere Application Server Administrator user that you are using for JMX notifications. Required if WebSphere security is enabled. .

*twss_user_pw* is the password for the user ID that you will use to access the Service Platform components and the Access Gateway from the PX Notification component Web service. Required if basic authentication is used to authenticate between the Service Platform components and the Access Gateway.

*wsadmin_options* are optional parameters that will be passed to the wsadmin command, which is invoked by the script. For example, if WebSphere global security is enabled, you can supply the user name and password: `-username wasadmin -password was123`

**Note:** If the silent option is provided, you must provide the passwords necessary for the setup. For example, you can use the database administrator password to create the tables. You must provide the passwords as command line arguments.

### Results

The First Steps script performs all of the following configuration steps:

- Initializes the necessary databases.
- Configures the common WebSphere Application Server resources.
- Configures the common Access Gateway resources.
- Configures the common Service Platform resources.
- Configures the resources for each installed Service Platform component and Web service implementation.
- Performs the same configuration for all server instances associated with the cluster.
- Configures the server and the node specific configuration items.

### What to do next

When the script completes, it might be necessary to perform some additional configuration, depending on which Web service implementations you plan to deploy. For more information, refer to the topic *Configuring connections for the Web service implementations*.

To check the results of running the First Steps script, refer to the topic *Verifying the completed configuration*.

## Configuring connections for the Web service implementations

After you run the First Steps script, additional configuration is required before some of the Web service implementations can be deployed. Use the WebSphere Integrated Solutions Console to perform this configuration.

If you are deploying Parlay-based Web service implementations, ensure that the Parlay Connector is configured to connect with the backend server. Use the procedures found in the *Configuring the Parlay Connector* section of this information center.

To configure other Web service implementations before deployment, use the following procedures as needed.

## Configuring SIP application composition on servers where Call Notification and Third Party Call are deployed

If you plan to run the Parlay X Call Notification over SIP/IMS and Parlay X Third Party Call over SIP/IMS Web service implementations on the same server (JVM), you must configure SIP application composition using the WebSphere Integrated Solutions Console.

### Before you begin

You must have completed the following steps:

- Completed the initial installation and configuration for TWSS base components and Web service implementations, which includes running the First Steps configuration script
- Started WebSphere Application Server, version 6.1.0.x or 7.0.0.1
- Started the deployment manager, node agents, and application servers where you have deployed the EAR files for the Parlay X Call Notification over SIP/IMS and Parlay X Third Party Call over SIP/IMS Web service implementations

### About this task

The JSR 116 standard for SIP applications states that multiple applications can be invoked for the same SIP request. The process of setting up applications to comply with this standard is called *application composition*.

To ensure that incoming SIP requests are handled properly when the Parlay X Call Notification over SIP/IMS and Parlay X Third Party Call over SIP/IMS Web service implementations run coresident on the same server, you must configure application composition for both applications.

For help in selecting the values to specify for application composition, refer to the topic *Setting up SIP application composition* in the WebSphere Application Server information center.

Complete the following steps to configure SIP application composition:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Applications** → **Enterprise Applications**.
3. In the Enterprise Applications list, click **PX21_CN_IMS**. This is the EAR file name for Parlay X Call Notification over SIP/IMS.
4. To specify the weight for the EAR file, click **Startup Behavior** and set the startup order.
5. Return to the Enterprise Applications list and click **PX21_CN_IMS**.
6. To specify the weight for the WAR files, click **Manage Modules** and set the startup order. The following Web modules exist for Call Notification:

   `callnotify-router`

   `callnotify-web`

   `CallNotificationSipServlet`
7. Return to the Enterprise Applications list and click **PX21_TPC_IMS**. This is the EAR file name for Parlay X Third Party Call over SIP/IMS.
8. To specify the weight for the EAR file, click **Startup Behavior** and set the startup order.
9. Return to the Enterprise Applications list and click **PX21_TPC_IMS**.
10. To specify the weight for the WAR file, click **Manage Modules** and set the startup order. The following Web module exists for Third Party Call: `ThirdParty Call Parlayx21 SIP`.
11. Click **Save** to save changes to the master configuration.
12. Restart the PX21_CN_IMS and PX21_TPC_IMS applications.

## Configuring WAP Push over SMPP

To make the WAP Push over SMPP Web service implementation ready for use, configure a primary and secondary backend server using the TWSS Administration Console.

### Before you begin

You must have completed the following steps:

- Completed the initial installation and configuration for TWSS base components and Web service implementations, which includes running the First Steps configuration script
- Started WebSphere Application Server, version 6.1.0.x or 7.0.0.1
- Started the deployment manager, node agents, and application servers where you have deployed the EAR file for this Web service implementation

### About this task

Complete the following steps to configure WAP Push over SMPP:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

   *port* is the secured port used to access the console. The default port is *9043*.

   **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services**.
4. Set the cluster in the scope as `Cell=DmgrCellname,Cluster=Service`.
5. Click **WAP10_PUSH_SMPP**
6. Create a new primary backend:
   a. Click **WAPPush Backends**.
   b. Click **New** to create a new primary backend.
   c. Type *`primary_backend_name`*: the name for the server where the application will attempt to send the messages first. This value must not use any special characters or contain spaces.
   d. Click **Add**.
7. Assign the connection properties to the primary backend:

   **Note:** For more information about these properties and their values, refer to the topic *Administering WAP Push over SMPP*.

   a. Click ***primary_backend_name***.
   b. Type the value for the **Host name**. This is the host name for the backend server.
   c. Type the value for the **Port**. This value is the port where the backend server listens.
   d. In the **System ID** field, type the user name to access the backend server.
   e. In the **Password** field, type the password for the user name.
   f. In the **System Type** field, supply the name of the system that requests the connection to the SMSC. Examples include `VMS` (voice mail system) and `OTA` (over-the-air activation system). This step may not be required for your SMSC, in which case you can either specify `NULL` or leave the field blank. The input string must be 12 characters or less.
   g. In the **Max Message Size** field, type the maximum size for messages that are sent to the backend server. The messages will be segmented if they are larger than the defined value. Set the value to zero to prevent segmentation. The default value is 245.
   h. In the **Max Targets Size** field, type the maximum size for a single target to be sent to the backend server. The default value is 255.
   i. In the **Confirmed Delivery** field, select `true` to receive delivery confirmation.
   j. In the **DataCoding** field, select a value (for example, `ISO-8859-8`) to define the encoding type.
   k. In the **Bind Mode** field, select a bind type to send to the SMSC (for example, `Transceiver(Trx)`).
   l. In the **Message Type** field, set the value to **SynchronousSMS**, **StatusLessSMS**, or **AsynchronousSMS**.
   m. Click **OK**.

8. Optional: Create a new secondary backend.
   a. Click **WAPPush Backends**.
   b. Click **New** to create a new secondary backend.
   c. Type `secondary_backend_name` for the name of the server where the application will attempt to send messages, if the primary server is not available.
   d. Click **Add**.
9. Assign the connection properties to the secondary backend:
   a. Click *secondary_backend_name*.
   b. Type the value for the **Host name**. This is the host name for the backend server.
   c. Type the value for the **Port**. This value is the port where the backend server listens.
   d. In the **System ID** field, type the user name to access the backend server.
   e. In the **Password** field, type the password for the user name.
   f. In the **System Type** field, supply the name of the system that requests the connection to the SMSC. Examples include `VMS` (voice mail system) and `OTA` (over-the-air activation system). This step may not be required for your SMSC, in which case you can either specify `NULL` or leave the field blank. The input string must be 12 characters or less.
   g. In the **Max Message Size** field, type the maximum size for messages that are sent to the backend server. The messages will be segmented if they are larger than the defined value. Set the value to zero to prevent segmentation. The default value is 245.
   h. In the **Max Targets Size** field, type the maximum size for a single target to be sent to the backend server. The default value is 255.
   i. In the **Confirmed Delivery** field, select `true` to receive delivery confirmation.
   j. In the **DataCoding** field, select a value (for example, `ISO-8859-8`) to define the encoding type.
   k. In the **Bind Mode** field, select a bind type to send to the SMSC (for example, `Transceiver(Trx)`).
   l. In the **Message Type** field, set the value to **SynchronousSMS**, **StatusLessSMS**, or **AsynchronousSMS**.
   m. Click **OK**.
10. Define the SMS Aliases:
   a. Click **SMS Alias Details**.
   b. Click **New**.
   c. Type **default** for the **Name**.
   d. Click **Add**.
   e. Click **default** and verify the alias is configured to communicate with the correct primary and secondary backend servers.
   f. Click **New**.
   g. Type ***device_1*** for the **Name**.
   h. Click **Add**.
   i. Click ***device_1*** and verify the alias is configured to communicate with the correct primary and secondary backend servers.
   j. Click **New**.

   k. Type ***device_2*** for the **Name**.
   l. Click **Add**.
   m. Click ***device_2*** and verify the alias is configured to communicate with the correct primary and secondary backend servers.
11. Configure values for **Services** and **WAPPush SMPP ENQUIRELINK and Bind Settings**. For details about these settings, refer to the topic *Administering WAP Push over SMPP*.
12. Configure ESMC settings for WAP Push. For details, refer to the topic *Configuring communication with the SMSC*.
13. Click **Save** to save changes to the master configuration.

## Configuring Parlay X SMS over SMPP

To make the Parlay X SMS over SMPP Web service implementation ready for use, configure a primary and secondary backend server using the TWSS Administration Console.

### Before you begin

You must have completed the following steps:

- Completed the initial installation and configuration for TWSS base components and Web service implementations, which includes running the First Steps configuration script
- Started WebSphere Application Server, version 6.1.0.x or 7.0.0.1
- Started the deployment manager, node agents, and application servers where you have deployed the EAR file for this Web service implementation

### About this task

Complete the following steps to configure Parlay X SMS over SMPP:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services** → **Clusters**.
4. Click ***cluster_name***.
5. Click **PX21_SMS_SMPP**.
6. Create a new primary backend:
   a. Click **SMS Backends**.
   b. Click **New** to create a new primary backend.

   c. Type *`primary_backend_name`*: the name for the server where the application will attempt to send the messages first. This value must not use any special characters or contain spaces.
   d. Click **Add**.
7. Assign the connection properties to the primary backend:
   a. Click *primary_backend_name*.
   b. Type the value for the **Host name**. This is the host name for the backend server.
   c. Type the value for the **Port**. This value is the port where the backend server listens.
   d. In the **System ID** field, type the user name to access the backend server.
   e. In the **Password** field, type the password for the user name.
   f. In the **System Type** field, supply the name of the system that requests the connection to the SMSC. Examples include `VMS` (voice mail system) and `OTA` (over-the-air activation system). This step may not be required for your SMSC, in which case you can either specify `NULL` or leave the field blank. The input string must be 12 characters or less.
   g. In the **Max Message Size** field, type the maximum size for messages that are sent to the backend server. The messages will be segmented if they are larger than the defined value. Set the value to zero to prevent segmentation. The default value is 254.
   h. In the **Max Targets Size** field, type the maximum size for a single target to be sent to the backend server. The default value is 255.
   i. In the **Confirmed Delivery** field, select `true` to receive delivery confirmation.
   j. In the **DataCoding** field, select a value (for example, `SMSC Default Encoding`) to define the encoding type.
   k. In the **Bind Mode** field, select a bind type to send to the SMSC (for example, `Transceiver(Trx)`).
   l. In the **Message Type** field, set the value to **SynchronousSMS**, **StatusLessSMS**, or **AsynchronousSMS**.
   m. Click **OK**.
8. Create a new secondary backend:
   a. Click **SMS Backends**.
   b. Click **New** to create a new secondary backend.
   c. Type *`secondary_backend_name`* for the name for the server where the application will attempt to send the messages if the primary server is not available. This value must not use any special characters or contain spaces.
   d. Click **Add**.
9. Assign the connection properties to the secondary backend:
   a. Click *secondary_backend_name*.
   b. Type the value for the **Host name**. This is the host name for the backend server.
   c. Type the value for the **Port**. This value is the port where the backend server listens.
   d. In the **System ID** field, type the user name to access the backend server.
   e. In the **Password** field, type the password for the user name.
   f. In the **System Type** field, supply the name of the system that requests the connection to the SMSC. Examples include `VMS` (voice mail system) and `OTA`

(over-the-air activation system). This step may not be required for your SMSC, in which case you can either specify `NULL` or leave the field blank. The input string must be 12 characters or less.

   g. In the **Max Message Size** field, type the maximum size for messages that are sent to the backend server. The messages will be segmented if they are larger than the defined value. Set the value to zero to prevent segmentation. The default value is 254.
   h. In the **Max Targets Size** field, type the maximum size for a single target to be sent to the backend server. The default value is 255.
   i. In the **Confirmed Delivery** field, select `true` to receive delivery confirmation.
   j. In the **DataCoding** field, select a value (for example, `SMSC Default Encoding`) to define the encoding type.
   k. In the **Bind Mode** field, select a bind type to send to the SMSC (for example, `Transceiver(Trx)`).
   l. In the **Message Type** field, set the value to **SynchronousSMS**, **StatusLessSMS**, or **AsynchronousSMS**.
   m. Click **OK**.

10. Define the SMS Aliases:
    a. Click **SMS Alias Details**.
    b. Click **New**.
    c. Type **default** for the **Name**.
    d. Click **Add**.
    e. Click **default** and verify the alias is configured to communicate with the correct primary and secondary backend servers.
    f. Click **New**.
    g. Type ***device_1*** for the **Name**.
    h. Click **Add**.
    i. Click ***device_1*** and verify that the alias is configured to communicate with the correct primary and secondary backend servers.
    j. Click **New**.
    k. Type ***device_2*** for the **Name**.
    l. Click **Add**.
    m. Click ***device_2*** and verify that the alias is configured to communicate with the correct primary and secondary backend servers.
11. Configure values for **Services** and **SMS SMPP ENQUIRELINK and Bind Settings**. For details about these settings, refer to the topic *Administering Parlay X SMS over SMPP*.
12. Configure ESMC settings for SMS over SMPP. For details, refer to the topic *Configuring communication with the SMSC*.
13. Click **Save** to save changes to the master configuration.

## Results

In conjunction with defining an alias, you can use policies to establish message priorities. Different aliases represent different accounts that are created in the Short Message Service Center (SMSC). Each account has a different priority and uses a unique user ID and password for the backend. Other details, such as the IP address and port number, are the same for all accounts.

Set the following policies to invoke priority :

**service.custom.priorityEnabled**
If the priority is set to *false*, then the priority check is not complete and no message priority will be set. However, if the priority is set to *true*, then the code will invoke a method which will override it.

**service.custom.requestPriorty**
Define aliases, and set a priority value for each alias. *1* is the alias with the highest priority.

**service.config.target.Aliases**
Supply a list of aliases to use for routing traffic to different SMSCs.

**service.config.target.Ranges**
Supply a list of target URI ranges.

For more information, refer to the topic *Default service policies for Parlay X SMS over SMPP*.

## Configuring Parlay X Terminal Location over MLP

To make the Parlay X Terminal Location over MLP Web service implementation ready for use, configure the MLP Connector using the TWSS Administration Console.

### Before you begin

You must have completed the following steps:

- Completed the initial installation and configuration for TWSS base components and Web service implementations, which includes running the First Steps configuration script
- Started WebSphere Application Server, version 6.1.0.x or 7.0.0.1
- Started the deployment manager, node agents, and application servers where you have deployed the EAR file for this Web service implementation

### About this task

Complete the following steps to configure Parlay X Terminal Location over MLP:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services**.
4. Set the cluster in the scope as `Cell=DmgrCellname,Cluster=Service`.

5. Click **PX21_TL_MLP.ear**.
6. In the Component Configuration page, click **MLP Connector**.
7. In the MLP Connector Settings page, configure the MLP connector by populating the following fields:
   a. Enter the location (URI) of the MLP location server in the **MLP Location Server Endpoint** field.
   b. If security is enabled at the MLP location server endpoint, select **Enable MLP Location Server security**. Then provide the user name and password.
   c. Select the appropriate **MLP protocol version**: 3.1 or 3.2.
   d. In the **MLP DTD** field, provide a location if the MLP Location server requires it in the request XMLs.
8. Click **Save** to save changes to the master configuration.

## Configuring Parlay X Multimedia Messaging over MM7

To make the Parlay X Multimedia Messaging over MM7 Web service implementation ready for use, configure a primary server (and optionally, a secondary backend) using the TWSS Administration Console.

### Before you begin

You must have completed the following steps:

- Completed the initial installation and configuration for TWSS base components and Web service implementations, which includes running the First Steps configuration script
- Started WebSphere Application Server, version 6.1.0.x or 7.0.0.1
- Started the deployment manager, node agents, and application servers where you have deployed the EAR file for this Web service implementation

### About this task

Complete the following steps to configure Parlay X Multimedia Messaging over MM7:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. Create a new primary backend for MMS:
   a. In the Navigation pane, click **Web Services** → **IMS Multimedia Messaging** → **Network Resources**.
   b. Click **MMS Backends**.

   c. Click **New** to create a new primary backend.
   d. Type *primary_backend_name* in the **Name** and **Value** fields. This value must not use any special characters or contain spaces.
   e. Click **Add**.
4. Assign the connection properties to the primary backend:
   a. Click ***primary_backend_name***.
   b. Type the value for the **Host name**. This is the host name for the backend server.
   c. Type the value for the **Port**. This value is the port on which the primary backend is running, for example 9080.
   d. Type the user name to access the backend server in the **System ID** field.
   e. Type the password for the user name in the **Password** field.
   f. Type the maximum size for messages that are sent to the backend server in the **Max Message Size** field. The messages will be segmented if they are larger than the defined value. Set the value to zero to prevent segmentation. The recommended value is 260.
   g. Type the maximum size for a single target to be sent to the backend server in the **Max Targets Size** field. The default value is 255.
   h. In the **Confirmed Delivery** field, select `false`.
   i. In the **Message Encoding** field, select `MMSC_Default_Encldoing` to define the encoding type.
   j. Click **OK**.
5. Optionally, create a new secondary backend:
   a. Click **MMS Backends**.
   b. Click **New** to create a new secondary backend.
   c. Type *`secondary_backend_name`* for the name for the server where the application will attempt to send the messages if the primary server is not available. This value must not use any special characters or contain spaces.
   d. Click **Add**.
6. Assign the connection properties to the secondary backend:
   a. Click ***secondary_backend_name***.
   b. Type the value for the **Host name**. This is the host name for the backend server.
   c. Type the value for the **Port**. This value is the port on which the secondary backend is running, for example 9080.
   d. Type the user name to access the backend server in the **System ID** field.
   e. Type the password for the user name in the **Password** field.
   f. Type the maximum size for messages that are sent to the backend server in the **Max Message Size** field. The messages will be segmented if they are larger than the defined value. Set the value to zero to prevent segmentation. The recommended value is 260.
   g. Type the maximum size for a single target to be sent to the backend server in the **Max Targets Size** field. The default value is 255.
   h. In the **Confirmed Delivery** field, select `false`.
   i. In the **Message Encoding** field, select `MMSC_Default_Encldoing` to define the encoding type.
   j. Click **OK**.
7. Define the MMS Alias:

    a. Click **MMS Alias Details**.
    b. Click **New**.
    c. In the **Name** field, type `default`.
    d. Click **Add**.
    e. Click **default** and verify that the alias is configured to communicate with the correct primary and secondary backend servers.
8. Click **Save** to save changes to the master configuration.

### Results

In conjunction with defining an alias, you can use policies to establish message priorities. Different aliases represent different accounts that are created in the Multimedia Message Service Center (MMSC). Each account has a different priority and uses a unique user ID and password for the backend. Other details, such as the IP address and port number, are the same for all accounts.

Set the following policies to invoke priority:

**service.custom.priorityEnabled**
For each alias you defined, set to `true` to enable priority checking.

**service.custom.requestPriorty**
For each alias, set a priority value. `1` is the alias with the highest priority.

**service.config.messaging.target.Aliases**
Supply a list of aliases to use for routing traffic to different MMSCs.

**service.config.messaging.target.Ranges**
Supply a list of target URI ranges.

For more information, refer to the topic *Default service policies for Parlay X Multimedia Messaging over MM7*.

## Verifying the completed configuration

After you perform the initial installation and configuration of Telecom Web Services Server, test the completed system to ensure that all connections are working properly.

### About this task

It is a good practice to test the data sources using the TWSS Administration Console and the Integrated Solutions Console.

To verify your configured system and prepare it for use, follow these steps.

1. Verify that the deployment manager and node agents are running.
2. Check the configuration log file for errors. The file is found in the following directory location: `*was_root*/installableApps/TWSS-Base/firststeps/logs`.
3. Use a Web service client application to send a test request to the Access Gateway.
4. Using a browser, test the URLs for the Web service implementations. You can check them against the list provided in the topic *Default Web service URIs*.
5. Optional: If you modified database authority so that the First Steps script could work with the database, you can now amend that authority using the following commands:

DB2

```
db2 connect to $dbAlias user $dbAdmin
db2 revoke DBADM on database from $dbUser
db2 terminate
```

Oracle

```
sqlplus $dbAdmin/$dbAdminPW%@$dbServer:$dbPort/$dbName as SYSDBA
revoke connect, resource, create table, create view, dba from $dbUser
```

where *dbUser* is the user ID to which you had granted authority.

6. In the Resources area of the Integrated Solutions Console, check the list of data sources that have been defined for your database or databases.

## WebSphere Application Server configuration for the Telecom Web Services Server components

After your Telecom Web Services Server (TWSS) installation is complete, the following WebSphere Application Server configuration values will have been set by default.

### Default WebSphere Application Server configuration values for the Service Policy Manager

The following default resources are defined for the Service Policy Manager:

- JAAS alias for *SPMAlias*
- Data source *jbdc/SPMDB*

**Note:** The data source is directed toward the TWSSDB database.

### Default WebSphere Application Server configuration values for the Access Gateway

The following default resources are defined for the Access Gateway:

- Work area for *TWSSPartition*
- Queue connection factory *TWSSSchedQCF* and destination for *jms/TWSSSchedQCF*
- Scheduler for *sched/TWSSSched*
- JMS queue for *jms/TWSSSchedQueue*
- Data sources:
  - *jbdc/MESSAGEDB* (data source for the Message Interceptor mediation primitive)
  - *jdbc/SLADB* (data source for the SLA Enforcement mediation primitive)
  - *jdbc/TWSSDB*

### Default WebSphere Application Server configuration values for the Service Platform components

The following default resources are defined for the Service Platform components:

- Service integration bus *TWSSBus*
- Queue connection factory *TWSSDefaultQCF* and destination for *jms/SP_PX21_PXNOTIFYQCF*
- JMS queue for *jms/SP_PX21_PXNOTIFYQueue*

- Activation specification for *eis/SP_PX21_PXNOTIFYAS*
- JAAS alias for *TWSSAlias*
- Data sources:
  - *jdbc/ADMINDB*
  - *jdbc/NMDB*
  - *jdbc/USAGEDB*
  - *jdbc/TWSSDB*

## Default WebSphere Application Server configuration values for the Web service implementations

The following default resources are defined for all of the Web service implementations:

- JAAS alias for *TWSSAlias* (for example, db2inst1)
- Data source *jdbc/TWSSDB*
- Service integration bus *TWSSBus* (as part of the configuration for the Service Platform components)

**Note:** Security is not enabled.

In addition, the following default resources are defined for specific Web service implementations:

**Parlay X SMS over SMPP**

- Queue connection factory *PX21_SMS_SMPPQCF* and destinations for *PX21_SMS_SMPPOutboundDest*, and *PX21_SMS_SMPPInboundDest*
- JMS queue for *jms/PX21_SMS_SMPPQueue* (outbound destination)
- JMS queue for *jms/PX21_SMS_SMPPCallbackQueue* (inbound destination)
- Activation specification for *eis/PX21_SMS_SMPPAS*
- Activation specification for *eis/PX21_SMS_SMPPCallbackAS*
- Connection factory for the SMS SMPP Connector resource adapter *PX21_SMS_SMPPJ2CCF*
- Activation specification for the resource adapter *eis/PX21_SMS_SMPPCallbackJ2CAS*

**WAP Push over SMPP**

- Queue connection factory *WAP10_Push_SMPPQCF* and destinations for *WAP10_Push_SMPPOutboundDest* and *WAP10_Push_SMPPInboundDest*
- JMS queue for *jms/WAP10_Push_SMPPQueue*
- JMS queue for *jms/WAP10_Push_SMPPCallbackQueue*
- JMS queue for *jms/WAP10_Push_SMPPQCF*
- Activation specification for *eis/WAP10_Push_SMPPAS*
- Activation specification for *eis/WAP10_Push_SMPPCallbackAS*
- Connection factory for the WAP SMPP Connector resource adapter *eis/WAP10_Push_SMPPJ2CCF*
- Activation specification for the resource adapter *eis/WAP10_Push_SMPPCallbackJ2CAS*

**Parlay X Multimedia Messaging over MM7**

- Queue connection factory *PX21_MM_MM7QCF* and destinations for *PX21_MM_MM7_OutboundDest* and *PX21_MM_MM7_InboundDest*

- JMS queue for *PX21_MM_MM7Queue*
- JMS queue for *jms/PX21_MM_MM7CallbackQueue*
- JMS queue for *jms/PX21_MM_MM7QCF*
- Activation specification for *eis/PX21_MM_MM7AS*
- Activation specification for *eis/PX21_MM_MM7CallbackAS*

**Parlay X SMS over Parlay**

- Queue connection factory *PX21_SMS_ParlayQCF* and destination *PX21_SMS_ParlayDest*
- JMS queue for *jms/PX21_SMS_ParlayQueue*
- JMS queue for *jms/PX21_SMS_ParlayQCF*
- Activation specification for *eis/PX21_SMS_ParlayAS*

## Adding a node or server Instance

If you have a running system with the cluster name previously defined in the configuration, and you need to add a node or server instance, you will need to publish the Telecom Web Services Server (TWSS) components to the new server instance. This is done by invoking the First Steps script with the `-syncServers` option specified.

### About this task

The processing of `-syncServers` makes use of the existing configuration and does not display the interactive configuration information window.

Follow these steps to publish the TWSS components to the new server instance:

1. On the server where Telecom Web Services Server base components have been installed, log in as *root* or with another valid ID.
2. Navigate to the following directory: `*was_root*/installableApps/TWSS-Base/firststeps`.
3. Edit the `firststeps.properties` file.

   Specify the scope for the operation by making sure that the name of the target server or cluster is specified for *each* of the following parameters:
   - AGTargetScope
   - SPMTargetScope
   - SPTargetScope

   In brief, the scope parameters can be specified as follows:
   - To configure a component on a cluster, type the cluster name in the appropriate target scope.
   - To configure a component on a server, either standalone or managed by the deployment manager, use the slash character–for example *`node/server`*.
   - To bypass configuration for the component altogether, type `None`.

   For more details about the configuration parameters and their values, refer to the topic *Performing the configuration silently*.

   **Note:** You can find copies of the original and new properties files in the logs *timestamp* directory. This applies to all nodes.
4. Perform the configuration by launching the First Steps script. Enter the command parameters on a single line:

```
./firststeps.sh @firststeps.properties -syncServers -DBServerAdminPass db_admin_pw -DBServerUserP
-WASAdminPass was_admin_pw -JMXAdminPass jmx_admin_pw -TWSSUserPass twss_user_pw wsadmin_options
```

Where:

`firststeps.properties` is the name of your saved configuration file.

*db_admin_pw* is the password for the Administrator user on the database server.

*db_user_pw* is the password for the user that will access the database server during runtime, for example `twssuser`.

*was_admin_pw* is the password for the WebSphere Application Server Administrator user.

*jmx_admin_pw* is the password for the WebSphere Application Server Administrator user that you are using for JMX notifications. Required if WebSphere security is enabled. .

*twss_user_pw* is the password for the user ID that you will use to access the Service Platform components and the Access Gateway from the PX Notification component Web service. Required if basic authentication is used to authenticate between the Service Platform components and the Access Gateway.

*wsadmin_options* are optional parameters that will be passed to the wsadmin command, which is invoked by the script. For example, if WebSphere global security is enabled, you can supply the user name and password: `-username wasadmin -password was123`

### Results

The TWSS components are published to the target server instance, or to the server instances in the target cluster.

The First Steps script performs all of the following configuration steps:

- Initializes the necessary databases.
- Configures the common WebSphere Application Server resources.
- Configures the common Access Gateway resources.
- Configures the common Service Platform resources.
- Configures the resources for each installed Service Platform component and Web service implementation.
- Performs the same configuration for all server instances associated with the cluster.
- Configures the server and the node specific configuration items.

## Installing updates for Telecom Web Services Server

Use the WebSphere Application Server Update Installer to install updates or fixes for the Telecom Web Services Server (TWSS) product.

### Before you begin

The WebSphere Application Server Update Installer must be installed on your system. Refer to the WebSphere Application Server documentation for the appropriate version level for the Update Installer.

### About this task

The information in this topic describes how to apply service, such as fix packs, to an existing installation of Telecom Web Services Server. For information about migrating to a new release of the product, refer to the topic *Migrating from a previous release*.

All updates or fixes for the WebSphere Telecom Web Services Server component are installed using the update installers associated with WebSphere Application Server version 6.1.0.x or 7.0.0.x.

To install TWSS updates, complete the following steps:

1. Install any required updates for the WebSphere Application Server product. To do this, shut down the application server, install the updates, and restart the server. The Readme file, included with every update or fix, identifies the updates that are required for WebSphere Application Server.
2. Install an IP Multimedia Subsystem plug-in file, in .pak format, which defines the WebSphere Telecom Web Services Server component to the update installer.
3. *Perform this step for each update or fix.* Install the update or fix, in .pak file format, using the update installer. This is a standard procedure. The Readme file contains detailed instructions.

   For additional information about the update installer, refer to the topic *Installing maintenance packages* in the WebSphere Application Server information center.
4. After the update or fix is installed, run the First Steps script with the update option. For additional information refer to the topic *Running the First Steps script interactively*.

# Removing Telecom Web Services Server

To remove Telecom Web Services Server, start by removing the applications. Then, optionally, drop the database tables and remove the databases. Finally, run the uninstallation program to remove all files that were installed during installation.

## Removing applications

Applications should be removed from WebSphere Application Server before you uninstall the Telecom Web Services Server (TWSS) components. Use the Integrated Solutions Console to stop the enterprise applications and remove them. After you have removed the applications, run the uninstallation program to remove the files from the server.

### About this task

You must remove all of the TWSS applications from the clusters. This includes all default message processing flows, all Service Platform components, all Web service implementations, and the Service Policy Manager. Because you are removing applications from all systems, you must complete the following steps on each deployment manager.

This task assumes that you are removing all services. If you are only removing select services, you may be able to skip some steps.

Complete the following steps to remove the applications:

1. Log in to the Integrated Solutions Console:

   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Click **Stop** to stop all of the applications.
3. Select the **check box** for all WebSphere Telecom Web Services Server applications.
4. Click **Uninstall**.
5. Click **OK**, and click **Save**.
6. Remove all resources created during the installation program. This includes, but is not limited to, all JDBC providers, data sources, authentication aliases, queues, scheduler, and buses.
7. Stop the servers, nodes, and deployment manager:
   a. Stop the servers. Run the following command:

      AIX
      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      Linux
      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

      The *was_profile_root* path contains the name of the application server profile (for example, AppSrv01).

      *server_name* is name of the application server.

      *user_name* represents your WebSphere Application Server administrator user ID.

      *password* represents the password associated with your *user_name*.

   b. Stop the nodes. Run the following command:

      AIX
      ```
      was_profile_root/bin/stopNode.sh -username user_name -password
      password
      ```

      Linux
      ```
      was_profile_root/bin/stopNode.sh -username user_name -password
      password
      ```

      ```
      was_profile_root/bin/stopNode.sh -username user_name -password
      password
      ```

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name.*

c. Stop the deployment manager. Run the following command:

AIX
```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

Linux
```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

```
was_profile_root/bin/stopManager.sh -username user_name -password
password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

8. Use the uninstallation program to remove the application files for the TWSS base components. If you installed the components in the default installation location, the uninstallation program is found in the following directory: *was_root*`/uninst_TWSS-Base`.
   a. Run the uninstallation program: `./uninstall`
   b. Review the information about the component to be removed, and click **Next**.
   c. Select the components you would like to uninstall, and click **Next**.
   d. Verify the correct components are being removed from the correct directory, and click **Next**.
   e. Click **Finish**.

   The uninstallation program removes the application files for the following components:
   - Access Gateway
   - Service Policy Manager
   - Service Platform components
   - TWSS Administration Console
   - Parlay Administration Console
   - First Steps script
9. Use the uninstallation program to remove the application files for the Web service implementations. If you installed the Web service implementations in the default installation location, the uninstallation program is found in the following directory: *was_root*`/uninst_TWSS-Services`.
   a. Run the uninstallation program: `./uninstall`
   b. Review the information about the component to be removed, and click **Next**.
   c. Select the components you would like to uninstall, and click **Next**.

   d. Verify the correct components are being removed from the correct directory, and click **Next**.
   e. Click **Finish**.

### Results

**Note:** AIX On AIX systems, the uninstallation programs do not remove the directories in which they are located. Therefore, you must delete the following directories manually: `/uninst_TWSS-Base`, `/uninst_TWSS-Services`.

## Removing databases and files

If you do not need to maintain the data, drop the database tables and remove the databases. Then run the uninstallation program to remove the database scripts for Telecom Web Services Server (TWSS) components and Web service implementations.

### About this task

This task assumes that you are removing all services. If you are only removing select services, you can skip some steps.

**Note:** You should also uninstall any deployed applications and then stop the server, before removing databases and uninstalling the installers.

Complete the following steps to remove the database tables and databases:

1. Log in to the database server as an administrator.
2. Drop the database tables.

   DB2 Issue the following command: `db2 DROP TABLE` *`table_name`*

   Oracle Using SQLPLUS, connect to the database and issue this command: `drop table` *`table_name`* `cascade constraints;`

   **Note:** If you are using Oracle 10.2.0.4 or later update level, 11 or later update level, DB/2 Enterprise Server Edition 9 5 FP1 or later and the recyclebin enabled, the database tables will not be dropped but will be renamed using the BIN$ string. To drop a table permanently, use the purge option. Refer to the Oracle documentation for details.
3. Remove the database.

   DB2 Select the database in the DB2 control panel, then select the drop action from the popup menu.

   Oracle Open the Database Configuration Assistant (DBCA) and issue the `drop` command.
4. Remove the files from the server using the uninstallation program. This uninstallation program removes the database initialization files for the following components:
   - Access Gateway
   - Service Policy Manager
   - Service Platform components

   a. Run the uninstallation program:
      ```
      ./uninstall
      ```

   b. Review the information about the component to be removed, and click **Next**.
   c. Select the components you would like to uninstall, and click **Next**.
   d. Verify the correct components are being removed from the correct directory, and click **Next**.
   e. Click **Finish**.
5. Remove the files from the server using the uninstallation program. This uninstallation program removes the database initialization files for the Web service implementations.

   If you installed the Web service implementations in the default installation location, the uninstallation program is found in the following directory: `*was_root*/uninst_TWSS-Services`.

   a. Run the uninstallation program:

      ```
      ./uninstall
      ```
   b. Review the information about the component to be removed, and click **Next**.
   c. Select the components you would like to uninstall, and click **Next**.
   d. Verify the correct components are being removed from the correct directory, and click **Next**.
   e. Click **Finish**.

### Results

**Note:** AIX On AIX systems, the uninstallation programs do not remove the directories in which they are located. Therefore, you must delete the following directories manually: `/uninst_TWSS-Base`, `/uninst_TWSS-Services`.

## Uninstalling the Trust Association Interceptor Security Component

The Trust Association Interceptor feature is uninstalled from the WebSphere Application Server Administration Console.

### About this task

Perform the following steps to uninstall the interceptor:

1. Delete the file `DHAIMSConnectorTAI.jar` from the *was_root*`/lib/ext` directory.

   **Note:** *was_root* is the installation root directory for WebSphere Application Server Network Deployment. By default, this directory is:

   AIX `/usr/IBM/WebSphere/AppServer`

   Linux `/opt/IBM/WebSphere/AppServer`

   `/opt/IBM/WebSphere/AppServer`
2. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

**Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
3. Click **Security** → **Global security** to display the Global security window.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Security** → **Secure administration, applications, and infrastructure**.
4. In the Global security window, under Authentication, click **Web and SIP security** → **Trust association**.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Web security** → **Trust association**.
5. Optional: In the Trust Association window, disable trust association:
   a. Clear the **Enable trust association** check box.
   b. Click **OK**.
   c. Click **Save**.
6. Return to the Trust Association window.
7. Under Additional Properties, click **Interceptors**.
8. Select **com.ibm.imsconnector.tai.HttpInterceptor** and **com.ibm.imsconnector.tai.SipInterceptor**.
9. Click **Delete**.
10. Click **Save**.
11. Restart the server.

## Uninstalling the Telecom Web Services Server base components using the interactive installer

You can use the interactive wizard provided with the base installer to uninstall the Telecom Web Services Server (TWSS) base components.

### About this task

To uninstall TWSS base components using the wizard, follow these steps.

1. Stop the servers, nodes, and deployment manager:
   a. Stop the servers. Run the following command:

AIX
```
was_profile_root/bin/stopServer.sh server_name -username
user_name -password password
```

Linux
```
was_profile_root/bin/stopServer.sh server_name -username
user_name -password password
```

```
was_profile_root/bin/stopServer.sh server_name -username
user_name -password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *was_profile_root* path contains the name of the application server profile (for example, AppSrv01).

*server_name* is name of the application server.

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

b. Stop the nodes. Run the following command:

AIX
```
was_profile_root/bin/stopNode.sh -username user_name -password password
```

Linux
```
was_profile_root/bin/stopNode.sh -username user_name -password password
```

```
was_profile_root/bin/stopNode.sh -username user_name -password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name.*

c. Stop the deployment manager. Run the following command:

AIX
```
was_profile_root/bin/stopManager.sh -username user_name -password password
```

Linux
```
was_profile_root/bin/stopManager.sh -username user_name -password password
```

```
was_profile_root/bin/stopManager.sh -username user_name -password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

2. Navigate to the following directory: `was_root/Uninstall_TWSS-Base`
3. Invoke the uninstall script to uninstall Telecom Web Services Server base components. Issue the following command:
```
./uninstall
```
4. Click **Next**.
5. In the Features panel, which displays a list of the components you have installed, select features you want to uninstall and click **Next**.
6. In the Summary Information page, click **Next** to start the uninstall process.
7. Optional: If the uninstall process reports any errors, verify that the uninstallation trace file is created at `was_root/logs/TWSS-Base_Uninstall.log`.

**Note:** The uninstall process may not remove the `TWSS-Base` directory under the `was_root/installableApps` directory. You can manually remove the directory if you no longer need it.

8. Manually remove the `*was_root*/license_TWSS` directory, which contains license files.

### Results

When the uninstallation has completed, check the appropriate logs in the *was_root*/`logs` directory. Refer to the topic *Monitoring log messages* for additional information about reading logs.

## Uninstalling Telecom Web Services Server base components silently

You can uninstall IBM WebSphere Telecom Web Services Server (TWSS) base components without user interactions. Instead, the uninstallation script uses a response file in which you supply options.

### About this task

A silent uninstallation uses a script to uninstall the product in silent mode, without the graphical user interface. Instead of displaying a script interface, the uninstallation program reads all of your responses from the response file which you provide.

To specify non-default options during a silent uninstallation, customize the response file.

1. Log in as root.
2. Navigate to the following directory: *was_root*/`Uninstall_TWSS-Base`
3. Customize the response file you used to install the TWSS base components. For more information, refer to the topic *Customizing the installation response file for Telecom Web Services Server base components*.
4. Select a umask that allows the owner to read and write to the files, and that allows others to access them according to the prevailing system policy.

   **Note:** For root, a umask of **022** is recommended. For non-root users an umask of **002** or **022** can be used, depending on whether or not the users share the group.
5. Invoke the uninstall script to uninstall Telecom Web Services Server components. The script makes use of your customized response file. Issue the following command:
   ```
   ./setup -i silent -f response_file_path
   where response_file_path
   ```
   is the full path to your response file.
6. Optional: Examine the uninstallation trace file at *was_root*/`logs/TWSS-Services_Uninstall.log` for any errors.

   **Note:** The uninstall process may not remove the `TWSS-Services` directory under the *was_root*/`installableApps` directory. You can manually remove the directory if you no longer need it.

### Results

Depending on the options selected, the uninstallation may take a while to complete. For example, it may take time for WebSphere Application Server to be stopped.

When the uninstallation has completed, check the appropriate logs in the *was_root*/`logs` directory. Refer to the topic *Monitoring log messages* for additional information about reading logs.

## Uninstalling the Telecom Web Services Server services using the interactive installer

You can use the interactive wizard provided with the service installer to uninstall the Telecom Web Services Server (TWSS) Web service implementations.

### About this task

To uninstall TWSS Web service implementations using the wizard, follow these steps.

1. Stop the servers, nodes, and deployment manager:
   a. Stop the servers. Run the following command:

      AIX
      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      Linux
      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      ```
      was_profile_root/bin/stopServer.sh server_name -username
      user_name -password password
      ```

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

      The *was_profile_root* path contains the name of the application server profile (for example, AppSrv01).

      *server_name* is name of the application server.

      *user_name* represents your WebSphere Application Server administrator user ID.

      *password* represents the password associated with your *user_name*.

   b. Stop the nodes. Run the following command:

      AIX
      ```
      was_profile_root/bin/stopNode.sh -username user_name -password
      password
      ```

      Linux
      ```
      was_profile_root/bin/stopNode.sh -username user_name -password
      password
      ```

      ```
      was_profile_root/bin/stopNode.sh -username user_name -password
      password
      ```

      **Note:** The `user_name` and `password` parameters are required only when security is enabled.

      Where:

      The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).

      *user_name* represents your WebSphere Application Server administrator user ID.

      *password* represents the password associated with your *user_name.*

   c. Stop the deployment manager. Run the following command:

      AIX
      ```
      was_profile_root/bin/stopManager.sh -username user_name
      -password password
      ```

```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

```
was_profile_root/bin/stopManager.sh -username user_name -password
password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

2. Navigate to the following directory: *`was_root`*`/Uninstall_TWSS-Services`
3. Invoke the uninstall script to uninstall Telecom Web Services Server services. Issue the following command: `./uninstall`
4. Click **Next**.
5. In the Features panel, which displays a list of the components you have installed, select features you want to uninstall and click **Next**.
6. In the Summary information page, click **Next** to start the uninstall process.
7. Optional: If the uninstall process reports any errors, verify that the uninstallation trace file is created at *`was_root`*`/logs/TWSS-Services_Uninstall.log`.

   **Note:** The uninstall process may not remove the `TWSS-Services` directory that is under the *`was_root`*`/installableApps` directory. You can manually remove the directory if you no longer need it.

## Uninstalling theWeb service implementations silently

You can uninstall IBM WebSphere Telecom Web Services Server (TWSS) Web service implementations without user interactions. Instead, the uninstallation script uses a response file in which you supply options.

### About this task

A silent uninstallation uses a script to uninstall the product in silent mode, without the graphical user interface. Instead of displaying a script interface, the uninstallation program reads all of your responses from the response file that you provide.

To specify non-default options during a silent uninstallation, customize the response file.

1. Log in as root.
2. Navigate to the following directory: *`was_root`*`/Uninstall_TWSS-Services`
3. Customize the response file you used to install the TWSS Web service implementations. For more information, refer to the topic *Customizing the installation response file for Telecom Web Services Server Web service implementations*.
4. Select a umask that allows the owner to read and write to the files, and that allows others to access them according to the prevailing system policy.

**Note:** For root, a umask of **022** is recommended. For non-root users an umask of **002** or **022** can be used, depending on whether or not the users share the group.

5. Invoke the uninstall script to uninstall TWSS components. The script makes use of your customized response file. Issue the following command:

   ```
   ./uninstall -silent -options full_path_to_response_file options
   ```

   where *options* can be any of the following:

   `-record` *`file_name`* generates an optional uninstall record file named *`file_name`*. This file records the values and options that were selected during the interactive uninstallation and can be used for subsequent silent installations.

   `-is:log trace.log -log !trace.log @ALL` produces a detailed log of the uninstall process. (Most installations will not require detailed logging.)

6. Optional: Verify the uninstallation by checking to see that the directories that contained files for the TWSS components–for example, *`was_root`*`/Uninstall_TWSS-Services`– have been removed.

## Results

Depending on the options selected, the uninstallation may take a while to complete. For example, it may take time for WebSphere Application Server to be stopped.

When the uninstallation has completed, check the appropriate logs in the *`was_root`*`/logs` directory. Refer to the topic *Monitoring log messages* for additional information about reading logs.

# Chapter 4. Configuring WebSphere Telecom Web Services Server

After you install and configure the product for the first time, it is ready to use. However, you may need to make further updates to the configured settings to accommodate your own particular needs. You may also need to update the settings later, as elements change in the network.

The Telecom Web Services Server Administration Console, an extension to the WebSphere Integrated Solutions Console, provides a convenient way to perform many of the configuration tasks for the various product components.

## Configuring the Access Gateway

You can configure the Access Gateway to facilitate its interaction with other Telecom Web Services Server components.

**Note:** For information about configuring WebSphere Performance Monitoring Infrastructure (PMI) indicators for the Access Gateway, refer to the topic *Performance metrics*.

### Connecting the Access Gateway cluster to the Web service implementations cluster

To configure the Access Gateway so that it can invoke Web service implementations, the First Steps script sets the `service.Endpoint` policy.

The `service.Endpoint` policy must be created for all requesters, for all operations, and for a specific service. You can define it for both the **all and unauthenticated** requesters, but it needs to be set in the scope of each Web service implementation.

When defining a policy, you must specify the policy scope in terms of the requester, service and operation in which the policy applies. By default, the service key for a specific service is the namespace of the interface, for example `http://www.csapi.org/wsdl/parlayx/terminal_location/v2_2/interface`. You can also use full Web service endpoints.

The requesters key utilizes constants to specify **all authenticated** or **all unauthenticated** requesters. The constant for all authenticated requesters is specified as **ALL**, and the constant for all unauthenticated requesters is specified as **unauthenticated**. Create the policy for all operations by specifying **ALL**. The Web service endpoint for the Web service implementation is created as the value of the `service.Endpoint` policy, for example:

```
http://localhost:9080/TWSS/ParlayX21/TerminalLocation/Parlay/services/TerminalLocation
```

It is good practice to check the `service.Endpoint` value to see that it is set as desired. The value should typically be a load balanced URL for the service in a cluster. To check the validity of the `service.Endpoint` value, paste it into the address bar of a Web browser and verify that it references the Web service you want.

Use the Service Policy Manager to set a policy. To create and update policies you can use the Service Policy Manager console, which provides a graphical user interface, or you can use the Web service operations createPolicies and updatePolices.

The First Steps script creates the necessary default policies. You can use the Service Policy Manager console to view or modify these policies.

## Configuring additional settings for the Access Gateway

You can configure name space bindings, which are used to provide configuration information to the Access Gateway. Some name space bindings provide integration points to other network elements while others provide runtime configuration. Depending on the requirements of your network, you can optionally configure several additional settings for the Access Gateway.

### Before you begin

To configure the optional name space bindings, you must have completed the following tasks:

- Installed WebSphere Enterprise Service Bus, Version 6.1.0.2
- Installed the Access Gateway
- Installed the Service Policy Manager
- Started the deployment manager, node agents, and application servers

### About this task

Complete the following steps on the deployment manager, and synchronize the changes with all nodes.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Environment** → **Naming** → **Name Space Bindings**.
3. To specify a scope, select the name of a cell in the drop-down list.
4. Optional: Create a name space binding for the attachment expiration policy:
   a. Click **New**.
   b. Click **String** for the binding type, and click **Next**.
   c. In the **Binding Identifier** field, type (or paste) `Attachment_expiration_policy`.
   d. In the **Name in Name Space** field, type `sca/attachments/ExpirationPolicy`.

   e. In the **String Value** field, specify the amount of time (in milliseconds) that a SOAP attachment should be stored in cache before it is deleted. The default value is `300000` (5 minutes).
   f. Click **Next**.
   g. Click **Finish**.
   h. Click **Save** to save changes to the master configuration.
5. Optional: To use a data source other than the default associated with the database that is caching the SOAP attachment content, create a name space binding for the database attachment cache:
   a. Click **New**.
   b. Click **String** for the binding type, and click **Next**.
   c. In the **Binding Identifier** field, type (or paste) `Database_attachment_cache`.
   d. In the **Name in Name Space** field, type `sca/attachments/DatabaseCache`.
   e. In the **String Value** field, specify the JNDI name of the data source that is used to connect to the database where you are caching the SOAP attachment content. If you are using a consolidated database topology, the recommended value is `jdbc/TWSSDB`. If you are using a distributed database topology, the recommended value is `jdbc/AGDB`.
   f. Click **Next**.
   g. Click **Finish**.
   h. Click **Save** to save changes to the master configuration.
6. Optional: To change the number of threads that are created to perform SOAP attachment cleanup, create a name space binding for the SOAP attachment cleanup thread count:
   a. Click **New**.
   b. Click **String** for the binding type, and click **Next**.
   c. In the **Binding Identifier** field, type (or paste) `SOAP_attachment_cleanup_thread_count`.
   d. In the **Name in Name Space** field, type `sca/attachments/CleanupThreadCount`.
   e. In the **String Value** field, specify the number of threads to be created to perform SOAP attachment cleanup. The default value is 1.
   f. Click **Next**.
   g. Click **Finish**.
   h. Click **Save** to save changes to the master configuration.
7. Optional: If you plan to send or receive SOAP attachments on request or response messages, you can limit the maximum size of the total attachment size, including all attachment parts. Configure the following name space binding:
   a. Click **New**.
   b. Click **String** for the binding type, and click **Next**.
   c. In the **Binding Identifier** field, type (or paste) `MaxSizeAttachments`.
   d. In the **Name in Name Space** field, type `sca/attachments/MaxSizeAttachments`.
   e. In the **String Value** field, specify the maximum size, in bytes, of all attachment parts. The value is an integer followed by the suffix `M`, `m`, `K`, `k`, `B`, or `b`. The default value is `1M` (1 megabyte).
   f. Click **Next**.
   g. Click **Finish**.

   - h. Click **Save** to save changes to the master configuration.
8. Optional: If you plan to send or receive SOAP attachments on request or response messages, you can limit the maximum size of any one attachment part. Configure the following name space binding:
   - a. Click **New**.
   - b. Click **String** for the binding type, and click **Next**.
   - c. In the **Binding Identifier** field, type (or paste) `MaxSizeAttachmentPart`.
   - d. In the **Name in Name Space** field, type `sca/attachments/MaxSizeAttachmentPart`.
   - e. In the **String Value** field, specify the maximum size, in bytes, for any single attachment part. The value is an integer followed by the suffix `M`, `m`, `K`, `k`, `B`, or `b`. The default value is `1M` (1 megabyte).
   - f. Click **Next**.
   - g. Click **Finish**.
   - h. Click **Save** to save changes to the master configuration.
9. Optional: You can enable the Access Gateway to process SOAP attachments in memory, rather than using disk I/O operations. Doing this could result in improved performance. Configure the following name space binding:
   - a. Click **New**.
   - b. Click **String** for the binding type, and click **Next**.
   - c. In the **Binding Identifier** field, type (or paste) `InMemoryProcessing`.
   - d. In the **Name in Name Space** field, type `sca/attachments/InMemoryProcessing`.
   - e. In the **String Value** field, specify `True` to enable the Access Gateway to process SOAP attachments in memory, or `False` to prevent the processing of SOAP attachments in memory. The default value is `True`. This default value is set for you when you run the First Steps script during initial setup of IBM WebSphere Telecom Web Services Server.
   - f. Click **Next**.
   - g. Click **Finish**.
   - h. Click **Save** to save changes to the master configuration.

### Configuring the edge server

Configure the LoadBalanceWeight tuning parameter in the IBM HTTP Server (IHS) `plugin-cfg.xml` file for the Access Gateway.

The **LoadBalanceWeight** option should be set to 1. This configures the desired `abcd` pattern.

**Note:** Configuring the edge server is dependent on the cluster topology of the actual setup. See the WebSphere Application Server information center for details about the configuration settings.

## Configuring the Service Platform and the Web service implementations

In many cases, the default setup for Telecom Web Services Server will provide the appropriate settings to facilitate connections between the Service Platform components, the Web service implementations, and other parts of the network. However, some modifications might be needed in some cases.

## Configuring endpoints for Service Platform components and Web service implementations

You can modify the default endpoint values for the Service Platform components and the Web service implementations if required by your network configuration.

### About this task

For Parlay-based Web service implementations, the Parlay gateway configuration is part of the configuration for the Parlay Connector component. It is not part of the configuration for the service implementations themselves.

Pay special attention to the following considerations:

- For Parlay X Terminal Status over SIP/IMS and Parlay X Presence over SIP/IMS, the presence server endpoint–for example IBM WebSphere Presence Server Component–must be configured. To configure the presence requests to go through an S-CSCF proxy, configure the endpoint to be the S-CSCF.
- For Parlay X Third Party Call over SIP/IMS, when connecting to an IMS network, the IMS S-CSCF endpoint must be configured. It functions as an outbound proxy.

To disable a Service Platform component for a specific Web service, blank out the endpoint URI. You can do this when you run the First Steps script during the initial setup of TWSS, or you can do it subsequently using the TWSS Administration Console.

To modify endpoints for a Service Platform component, follow these steps:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Select the TWSS Administration Console.
3. Click **Web services**.
4. Click the name of a deployed Web service.
5. Click the name of the Service Platform component for which you want to change the endpoints, and do one of the following:
   - To change an endpoint, substitute the new URI for the one that is displayed.
   - To disable a Service Platform component for a specific Web service, blank out the endpoint URI.

## Configuring the Parlay Connector

The Parlay Connector provides connectivity between a Parlay application and a Parlay gateway. The connector converts Java, the language of the application, to CORBA, the language of the gateway.

If you intend to use any of the Parlay-based Web service implementations, you are responsible for providing a way for them to connect to a Parlay gateway. Both the connector and the gateway must support the same version of the Parlay specification.

Use the Parlay Administration Console to configure the Parlay Connector.

Parlay is a set of object-oriented application programming interfaces (APIs) created by the Parlay Group to provide a standard way of interfacing between enterprise-based client applications and a telecommunications network.

The APIs enable a Parlay application to interact with the public switched telephone network (PSTN). An application developer is able to develop applications and components, such as messaging, mobility, and end-to-end quality of service, independent of the underlying voice and multimedia network.

A Parlay application typically runs on an enterprise server, such as WebSphere Application Server, and communicates through Common Object Request Broker Architecture (CORBA) with a Parlay gateway. The Parlay gateway receives requests from the application and performs the corresponding actions with the telecommunications network on behalf of the application. The gateway also notifies Parlay applications when significant events occur in the network.

Parlay applications use an interface definition language (IDL) file which contains Parlay definitions used to construct many of the programming interfaces used in Parlay applications. Some aspects of the Parlay definition are beyond the scope of the API. You might need to customize your gateway to ensure that you can integrate the gateway successfully and validate that the gateway works correctly.

### Parlay Connector templates

Parlay Connector templates are used to configure the Parlay Connector. The templates allow you to create or change the Parlay Connector configuration for an application EAR file.

The Parlay Connector is implemented as a managed object. The Parlay Connector obtains service managers for the client application. You must configure the Parlay Connector to locate and connect to the Parlay gateway to obtain service managers from the Parlay gateway. You can create or change this configuration.

After you create a configuration, the configuration status is inactive. When you open the Parlay Connector, the connector attempts to locate an active configuration. If an active configuration is found, the connector uses the information in the active configuration for initialization. If no active configuration is found, the connector uses the information in the default template.

Using a template does not restrict the data that you can specify in a configuration. If you use the default template, your configuration is created with settings for a session with no Parlay gateway connection. After it is created, however, you can update the configuration to connect to any Parlay gateway. Templates are provided for individual Parlay gateways, which minimizes the need for updates.

### Multiple configurations

Multiple configurations can be defined for the Parlay Connector. The behavior of the configuration set is as follows:

- An administrator must activate a configuration in order for the Parlay Connector to use the configuration. Only one configuration can be active at a time. You can activate and then deactivate the configuration to use a different configuration.
- After the application is started, the administrator cannot select a new active configuration for the application. You must stop and restart the application.
- You can deactivate an active configuration to enable the connector to restart without connecting to the gateway.

Under some circumstances, updates to the Parlay Connector configurations might not be displayed when the application is refreshed. If this occurs, restart WebSphere Application Server, the WebSphere Application Server cluster, or the Rational® Application Developer test environment where the EAR file is deployed. When the server restarts, open the application to view the changes.

**Creating a new Parlay Connector configuration:**

When you first open the Parlay Connector for an application, no configurations are defined. You must use a configuration template to create the first Parlay Connector configuration. To create a new configuration, copy an existing configuration or use a template.

**Before you begin**

You must have created all required databases.

**About this task**

Complete these steps to create a new configuration for the Parlay Connector. If you use a Parlay simulator installed on the host where the application runs, you might need to create a new configuration using the template provided for your simulator.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.

6. Select a configuration template or select an existing configuration. To connect to a Parlay gateway or to a simulator not listed, select the **Generic gateway template** and customize it as needed.
7. Click **New**.
8. Type a name for the configuration.
9. Click **OK**.

**Changing a Parlay Connector configuration:**

Existing Parlay Connector configurations can be changed as part of configuring the Parlay Connector.

**About this task**

Complete these steps to change an existing configuration for the Parlay Connector. If you use a Parlay simulator installed on the host where the application runs, you might need to customize the configuration using the template provided for your simulator.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of the configuration and customize it as needed.
7. Click **OK**.

**Activating and deactivating a Parlay Connector configuration:**

When you have your own Parlay Connector configurations, use the Integrated Solutions Console to activate and deactivate them. When started, the application uses the default configuration for the Parlay Connector and does not attempt to connect to a Parlay gateway or simulator.

**About this task**

Complete these steps to activate or deactivate a Parlay Connector configuration.

1. Log in to the Integrated Solutions Console:

   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)

   c. Click **Log in**.

2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Select the **check box** for the configuration you want to activate or deactivate.
7. Click one of these options.
   - **Activate** to activate the selected configuration. If a configuration was already active before you made this change, the previously active configuration becomes inactive.
   - **Deactivate** to deactivate the configuration.
8. Restart the application.

## Gateway connections

The Parlay specification does not define the process of locating the Parlay gateway, but it does provide service type information and interface definitions to be used when configuring the gateway location.

The Parlay Connector uses configuration data to control dynamic loading of the gateway location class. You can configure the connector to accept the default gateway location class or to use a customized gateway location class. You can configure the default gateway location class for some customization during the location process. If a customized gateway location class is used, it will be passed a hash table containing the Parlay Connector configuration and state information.

### Configuring for gateway failover

For failover, you can configure a secondary Parlay gateway as a backup in case the primary gateway fails. For each additional gateway, you must configure unique service IDs.

The Parlay-based Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway.

Note that if the primary gateway fails and the secondary, or backup, gateway assumes the workload, the session IDs and assignment IDs for each gateway must be unique or an exception is generated.

**Configuring the object request broker:**

To resolve the incompatibility of the ORB code set, the object request broker (ORB) has to be configured.

**Before you begin**

You must start the WebSphere Application Server.

**About this task**

Complete these steps to configure the ORB.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Expand **Servers**.
4. Click **Application Servers**.
5. Click the name of the server.
6. In the Container Settings section, expand **Container Services** and click **ORB Service**.
7. In the Additional Properties section, click **Custom Properties**.
8. Optional: Some vendors recommend configuring a specific character encoding to facilitate connections through the gateway. Define a new custom property by performing these steps:
   a. Click **New**.
   b. Type `com.ibm.CORBA.ORBCharEncoding` in the **Property name** field.
   c. Type `ISO8859_1` in the **Property value** field.
   d. Click **Add**.
9. Optional: To change a custom property, type the new value in the appropriate row of the Property value column.
10. Click **OK**.
11. Restart the WebSphere Application Server.
12. Click **Republish**.

**Configuring a remote gateway:**

The Parlay Connector can be configured for a remote Parlay gateway.

**Before you begin**

You must start the WebSphere Application Server.

**About this task**

Complete these steps to configure the Parlay Connector for a remote gateway.

1. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
    b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    c. Click **Log in**.
2. In the navigation pane, expand **Servers**.
3. Expand **Server Types**.
4. Click **WebSphere application servers**.
5. Click the name of the server.
6. In the Communications section, expand **Ports**.
7. Click **Details**.
8. Click **BOOTSTRAP_ADDRESS**.
9. Type a value in the **Host** field.
10. Type a value in the **Port** field.
11. Click **OK** .
12. Restart WebSphere Application Server.
13. Click **Republish**.

**Gateway location servers:**

Gateway location servers provide information to the Parlay Connector needed to make initial contact with the Parlay gateway.

The gateway location server publishes the necessary information for making contact which is looked up using an internet object reference (IOR), a uniform resource locator (URL) string, or an IP address and port. If the IOR or URL is for the Parlay gateway initial reference (IpInitial), the Parlay Connector can use it to make initial contact with the gateway. If the published IOR or URL is for the Parlay gateway name service, the connector can use either the IOR or the URL with the configured name service key to lookup the initial contact with the gateway. It works the same way with an IP and a port.

The default Parlay Connector locator software uses certain methods for providing its IpInitial to clients. A gateway that publishes its IpInitial IOR or URL to an IP address and port does not need customized locator software. A gateway that registers its IpInitial with a name service, and then publishes the name service IOR or URL to an IP address and port also does not need customized locator software.

If a gateway uses some other method to provide its IpInitial to clients, customized locator software is needed before the gateway location server can provide initial contact information to the Parlay Connector.

**Configuring the connection with the gateway:**

You must determine if the configuration data for the Parlay Connector includes information needed to connect with the Parlay gateway. If the Parlay gateway connection information is included, the Parlay Connector must locate the gateway and make the initial connection.

**About this task**

Complete these steps to configure the Parlay Connector to connect to the Parlay gateway.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. Select a value from the **Gateway location method** drop-down list.
   - configured_ipinitial_locator
   - published_ipinitial_locator
   - configured_nameservice_locator
   - published_nameservice_locator
   - other
8. In the Additional Properties section, select the same option as the one previously selected from the **Gateway location method** drop-down list.
9. In the Additional Properties section, click **Gateway location**.
10. Type a value for each field displayed. The fields displayed are based on the value specified for **Gateway location method** . Possible configuration items are:
    - Gateway NameService publisher - host
    - Secondary gateway NameService publisher - host
    - Gateway NameService publisher - port

- Secondary gateway NameService publisher - port
- NameService key for IpInitial
- Gateway IpInitial publisher host
- Secondary gateway IpInitial publisher host
- Gateway IpInitial publisher port
- Secondary gateway IpInitial publisher port
- NameService IOR/URL
- Secondary NameService IOR/URL
- IpInitial IOR/URL
- Secondary IpInitial IOR/URL

11. Type a value in the **Connection retries** field. The default value is `20`. Custom software might ignore this field or might use this field for some other purpose.
12. Type a value in the **Connection retry interval in seconds** field. The default value is `2`. Custom software might ignore this field or might use this field for some other purpose.
13. Optional: Type a value in the **Custom gateway locator classname** field if you need a customized gateway locator class for your target Parlay gateway. If you use the default custom gateway locator implementation, you do not need to type a value in this field. The default value is `none`.
14. Click **OK** to save your changes.

*Configured IP initial locator:*

The configured_ipinitial_locator method locates the Parlay gateway by means of the Internet Object Reference (IOR) or the Uniform Resource Locator (URL), for its initial reference object (IpInitial) supplied in the Parlay Administration Console.

**Purpose**

The default GatewayLocator uses the IOR or the URL string for the IpInitial object to locate the Parlay gateway.

**Parameters**

**IpInitial IOR/URL**

This field specifies the IOR or the URL for the gateway initial reference (IpInitial). There is no default value for this field. Specify a value similar to this example.

```
IOR:1234567890ABCDEFEDCBA09876543211234567890ABCDEFEDCBA0987654321
corbaloc::9.124.23.128:2050/parlayPOA/IpInitial
```

**Secondary IpInitial IOR/URL**

This field specifies the IOR or the URL for the gateway secondary initial reference (IpInitial). There is no default value for this field. The default gateway locator class for the Parlay Connector ignores the data in this field if it is able to obtain the gateway initial reference using the value specified in IpInitial IOR/URL. Custom software might ignore this field or use this field for some other purpose. Specify a value similar to this example.

```
IOR:1234567890ABCDEFEDCBA09876543211234567890ABCDEFEDCBA0987654321
corbaloc::9.124.23.128:2050/parlayPOA/IpInitial
```

*Published IP initial locator:*

The published_ipinitial_locator method locates the Parlay gateway if the gateway vendor provides a host and a port where the Internet Object Reference (IOR) or the Uniform Resource Locator (URL) for its initial reference (IpInitial) object can be obtained.

**Purpose**

The default GatewayLocator connects to the configured IP address and port using a socket and reads the IOR or the URL string of the Parlay gateway initial reference (IpInitial). This allows a gateway location server to be built and run outside of a WebSphere process, and facilitates the integration of the Parlay gateway with the Parlay Connector.

**Parameters**

**Gateway IpInitial publisher host**
This field specifies the server where the IOR or the URL for the gateway IpInitial information is published. There is no default value for this field. The value specified must be either the IP address of the host or the fully qualified host name. Specify a value similar to this example.

```
123.456.789.1
dev01.yourcompany.com
```

**Gateway IpInitial publisher port**
This field specifies the port number of the server where the IOR or the URL for the gateway IpInitial information is published. The port number must be the port number for the host specified in Gateway IpInitial publisher host. There is no default value for this field. Specify a value, for example `7111`.

**Secondary gateway IpInitial publisher host**
This field specifies the secondary server where the IOR or the URL for the gateway IpInitial information is published. There is no default value for this field. The default locator software ignores this field unless it is unable to obtain the gateway's IpInitial reference using the values defined for Gateway IpInitial publisher host and Gateway IpInitial publisher port. Custom software might ignore this field or might use this field for some other purpose. The value specified must be either the IP address of the host or the fully qualified host name. Specify a value similar to this example.

```
123.456.789.1
dev01.yourcompany.com
```

**Secondary Gateway IpInitial publisher port**
This field specifies the port number of the secondary server where the IOR or the URL for the gateway IpInitial information is published. There is no default value for this field. The port number must be the port number for the host specified in Secondary Gateway IpInitial publisher host. The default locator software ignores this field unless it is unable to obtain the gateway's IpInitial reference using the values defined for Gateway IpInitial publisher host and Gateway IpInitial publisher port. Custom software might ignore this field or might use this field for some other purpose. Specify a value, for example `23104`.

*Configured name service locator:*

The configured_nameservice_locator method locates the Parlay gateway using the Internet Object Reference (IOR) or a Uniform Resource Locator (URL) for its name service. This value is specified in the Parlay Administration Console.

**Purpose**

The default GatewayLocator uses the configured IOR or the URL string for the name service to locate the Parlay gateway.

**Parameters**

**NameService IOR/URL**
This field specifies the IOR or the URL for the gateway ORB NameService. There is no default value for this field. Specify a value similar to this example.

```
IOR:1234567890ABCDEFEDCBA09876543211234567890ABCDEFEDCBA0987654321
corbaloc::9.124.23.128:2050/StandardNS/NameServer-POA/_root
```

**Secondary NameService IOR/URL**
This field specifies the IOR or the URL for the gateway ORB secondary NameService. The default gateway locator class for the Parlay Connector ignores the data in this field if it is able to obtain the gateway initial reference using the value specified in NameService IOR/URL. There is no default value for this field. Custom software might ignore this field or use this field for some other purpose. Specify a value similar to this example.

```
IOR:1234567890ABCDEFEDCBA09876543211234567890ABCDEFEDCBA0987654321
corbaloc::9.124.23.128:2050/StandardNS/NameServer-POA/_root
```

**NameService key for IpInitial**
This field specifies a string defined by the Parlay gateway. This string is used as the key when the gateway location method queries the gateway NameService to obtain the initial interface (IpInitial) for the gateway. The default value for this field is the value provided for the **Authentication domain ID**. Custom software might ignore this field or use this field for some other purpose. Specify a value similar to this example.

```
parlayPOA/IpInitialParlay
IpInitial
```

*Published name service locator:*

The published_nameservice_locator method locates the Parlay gateway if the gateway vendor provides a host and a port where the Internet Object Reference (IOR) or the Uniform Resource Locator (URL) for its name service can be obtained.

**Purpose**

The default GatewayLocator uses the configured IOR or the URL string for the name service to locate the Parlay gateway. This allows a gateway location server to be built and run outside of the WebSphere process and facilitates the integration of the Parlay gateway with the Parlay Connector.

**Parameters**

**Gateway NameService publisher host**
This field specifies the server where the IOR or the URL for the gateway ORB name service is published. There is no default value for this field. The value specified must be either the IP address of the host or the fully qualified host name. Specify a value similar to this example.

```
123.456.789.1
dev01.yourcompany.com
```

**Gateway NameService publisher port**
This field specifies the port number of the server where the IOR or the URL for the gateway ORB name service is published. The port number must be the

port number for the host specified in Gateway NameService publisher host. There is no default value for this field. Specify a value, for example 7111.

**Secondary Gateway NameService publisher host**

This field specifies the secondary server where the IOR or the URL for the gateway ORB name service is published. There is no default value for this field. The default locator software ignores this field unless it is unable to obtain the gateway ORB name service reference using the values defined for Gateway NameService publisher host and Gateway NameService publisher port. Custom software might ignore this field or might use this field for some other purpose. The value specified must be either the IP address of the host or the fully qualified host name. Specify a value similar to this example.

```
123.456.789.1
dev01.yourcompany.com
```

**Secondary Gateway NameService publisher port**

This field specifies the port number of the secondary server where the IOR or the URL for the gateway ORB name service is published. The port number must be the port number for the host specified in Gateway NameService publisher host. There is no default value for this field. Custom software might ignore this field or might use this field for some other purpose. Specify a value, for example 23104.

**NameService key for IpInitial**

This field specifies a string defined by the Parlay gateway. This string is used as the key when the gateway location method queries the gateway NameService to obtain the initial interface (IpInitial) for the gateway. The default value for this field is the value provided for the **Authentication domain ID**. Custom software might ignore this field or use this field for some other purpose. Specify a value similar to this example.

```
parlayPOA/IpInitialParlay
IpInitial
```

## Authentication and encryption

The Parlay Connector authenticates with the Parlay gateway to ensure that is a valid client for the gateway and to ensure that the gateway is a valid server for the Parlay Connector. You can configure the gateway to use encryption during the authentication process.

After the Parlay Connector has obtained the initial access object (IpInitial) of the Parlay gateway, the Parlay Connector must authenticate with the gateway to ensure that it is an authorized client of the Parlay gateway. The connector also authenticates with the gateway to ensure that the gateway is a valid server for the connector.

The Parlay 3.x authentication method has been upgraded to the Parlay 4.x authentication method sequence with backward compatibility to the 3.x method. Parlay 3.x and 4.x services are supported. The authentication process uses the standard Parlay APIs defined by the Parlay Group. The Parlay specification defines the authentication processing flow however, procedures for applying encryption during authentication may vary.

The Parlay 4.x Authentication process changes the Encryption mechanisms to Hashing mechanisms. Hashing methods produces a one way digest which will be used as digital signatures and sent to the other party. The party who has initiated the challenge will also generate a signature, using the agreed Authentication mechanism (Hashing mechanism). A comparison of both these signatures completes the Authentication procedure.

The connector uses configuration data to control dynamic loading of the gateway authentication classes. This enables you to accept the default gateway authentication classes or to configure customized gateway authentication classes.

If the Parlay gateway requires that different key sets be used, you need custom software either for authentication or for signing service agreements. Configure the additional keys as custom parameters for the customized software. If no other reason exists for custom software for these processes, implement a custom authenticator class and a custom authenticator callback class.

You might configure customized authentication classes if the default implementations cannot be configured to authenticate with your Parlay gateway. During the authentication process, you can configure the default gateway authentication classes for customization. If you use customized gateway authentication classes, they are passed as a hash table containing the Parlay Connector configuration and state information.

**Authentication mechanisms:**

The Parlay 4.2 level framework specification provides support for choosing a framework version with which to authenticate. This can be useful where a client requires services that depend on the framework version. Backward compatibility with the version 3.x authentication scheme is provided for all configurations.

Additional classes for OAS, Lucent, and Aepona have been provided for Parlay version 4.2 authentication support. This version is shown as a configurable item in the administration console for all the configurations.

By comparing the digests at both the gateway end and the connector end, you can tell whether the authentication has succeeded. Hashing methods proposed by the Parlay group, such as P_OSA_MD5, P_OSA_HMAC_SHA1_96 and P_OSA_HMAC_MD5_96, are included.

**Note:** Refer to the specifications at http://www.3gpp.org, for more information about keys and signing algorithms for the hashing methods.

Parlay gateway authentication includes three processes:

- The Parlay Connector initiates authentication processing and negotiates the encryption method to be used when forming the response to a challenge in the current session. The Parlay Connector object that performs this step is the Authenticator.
- The Parlay gateway authenticates the Parlay Connector by calling the Parlay Connector with an authentication challenge and verifying that the Parlay Connector responds correctly. The Parlay Connector object that performs this step is the Authenticator Callback.
- The Parlay Connector authenticates the Parlay gateway by calling the Parlay gateway with an authentication challenge and verifying that the Parlay gateway responds correctly. The Parlay Connector object that performs this step is the Authenticator.

  **Note:** A customized plug-in class can override either or both of the Authenticator and Authenticator Callback.

**Default Parlay gateway authentication**

The default implementation of the Authenticator interface begins the authentication process.

The encryption method waits for the Parlay gateway to call the Authenticator Callback object. When the Parlay gateway has notified the Authenticator Callback that it has successfully authenticated itself, the Authenticator object sends a random byte array to the Parlay gateway. The Parlay gateway must encrypt this data and return the result to the Authenticator. If the Authenticator verifies the response, then the Authenticator notifies the Parlay gateway that it has successfully authenticated itself.

The default implementation of the Authenticator Callback interface receives a byte array from the Parlay gateway. The Authenticator Callback encrypts this data and returns the result to the Parlay gateway. If the Parlay gateway verifies the response, then the Parlay gateway notifies the Authenticator Callback that it has successfully authenticated itself.

**Parlay gateway authentication for plug-ins**

If customer support is needed for authenticating with a Parlay gateway, either the Authenticator class name configuration field or the Authenticator callback class name configuration field, or both, can be used to specify custom classes.

The class specified in the Authenticator class name configuration field must implement the `com.ibm.wast.parlay.connector.Authenticator` interface.

The class specified in the Authenticator callback class name configuration field must implement the `com.ibm.wast.parlay.api.fw.fw_access.trust_and_security.IpClientAPILevelAuthentication` and `com.ibm.wast.parlay.connector.AuthenticatorCallback` interfaces.

### Configuring encryption:

If you configured the Parlay Connector to use encryption during authentication or while signing service agreements, you must also configure the data needed for the configured encryption processing. The data includes encryption keys.

**Before you begin**

If the Parlay gateway requires that different key sets be used, you need custom software either for authentication or for signing service agreements. Configure the additional keys as custom parameters for the customized software.

**About this task**

Complete these steps to configure the Parlay Connector for encryption:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

*port* is the secured port used to access the console. The default port is *9043*.

**Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Encryption**. Because encryption keys are confidential, you cannot see the data that you enter in some fields.
8. Type a value in each field for which you want to define an encryption key for the Parlay Connector.

**none** No encryption.

**DES/56 key**
The key to use for 56-bit Data Encryption Standard (DES) encryption. A string of 16 hexadecimal digits. If not specified, the Parlay Connector does not support 56-bit DES encryption.

**TripleDES (DES/128) key**
The key to use for 168-bit Triple Data Encryption Standard (Triple DES) encryption. A string of 48 hexadecimal digits. If not specified, the Parlay Connector does not support Triple DES encryption.

**Connector RSA/512 private key**
The private key to use for 512-bit Rivest, Shamir, and Adleman (RSA) encryption. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. The value of this key is known by the Parlay Connector, but not by the Parlay gateway.

**Connector RSA/512 public key**
The public key to use for 512-bit RSA encryption. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. The value of this key is known by the Parlay gateway.

**Connector RSA/1024 private key**
The private key to use for 1024-bit RSA encryption. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. The value of this key is known by the Parlay Connector, but not by the Parlay gateway.

**Connector RSA/1024 public key**
The public key to use for 1024-bit RSA algorithm with 1024-bit keys. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during

authentication processing and for signing service agreements. The value of this key is known by the Parlay gateway.

**Gateway RSA/512 public key**
The Parlay gateway's public key to use for 512-bit RSA encryption. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. The value of this key is known by the Parlay gateway. Use this field to configure the Parlay gateway public key if it is not given to you in a security certificate.

**Gateway RSA/512 security certificate**
The certificate containing the Parlay gateway's public key to use for 512-bit RSA encryption. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. Use this field to configure the Parlay gateway RSA/512 public key if it is given to you in this form.

**Gateway RSA/1024 public key**
The Parlay gateway's public key to use for 1024-bit RSA encryption. Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. The value of this key is known by the Parlay gateway. Use this field to configure the Parlay gateway public key if it is not given to you in a security certificate.

**Gateway RSA/1024 security certificate**
The certificate containing the Parlay gateway's public key to use for 1024-bit RSA encryption. . Two strings of hexadecimal digits–the key modulus and the key exponent–separated by the pound sign (#). Might be used during authentication processing and for signing service agreements. Use this field to configure the Parlay gateway RSA/1024 public key if it is given to you in this form.

**Gateway 'no encryption' constant**
The string used by the Parlay gateway to specify that no encryption will be used. Use this field if the Parlay gateway does not recognize the empty string.

**Shared secret**
A shared secret known to both the Parlay Connector and the Parlay gateway.

**Secret keys**:

- **OSA_HMAC_SHA1_96** - The secret key that is shared by the Parlay Connector and the Parlay gateway, which will be hashed using OSA_HMAC_SHA1_96. A string that is used during the start of the authentication in Parlay 4.x. (OAS treats this as a hexadecimal string and Lucent treats it as a text string.)
- **OSA_HMAC_MD5_96** - The secret key that is shared by the connector and the gateway, which will be hashed using OSA_HMAC_MD5_96. A string that is used during the start of the authentication in Parlay 4.x. (OAS treats this as a hexadecimal string and Lucent treats it as a text string.)

**Private key**:

- **SHA1_DSA** - The private key used by the Parlay Connector to decrypt the service agreement signed by the Parlay gateway, using the P_SHA1_DSA algorithm. A string with four parts, separated by pound signs (#).

  **Public key**:

  - **SHA1_DSA** - The public key used by the Parlay Connector to encrypt the service agreement sent to the Parlay gateway, using the P_SHA1_DSA algorithm. A string with four parts, separated by pound signs (#).

9. Click **OK** to save your changes.

**Configuring authentication:**

The Parlay Connector must authenticate with the Parlay gateway to ensure that it is an authorized client of the Parlay gateway. The Parlay Connector also authenticates with the Parlay gateway to ensure that it is connected to the correct gateway.

**Before you begin**

- Ensure that either the Parlay 3x or 4x authentication method is used. Unless **4x** is specified, the authentication method is defaulted to Parlay 3x.
- Configure encryption keys if the Parlay Connector uses encryption during the authentication process.

**About this task**

Complete these steps to configure the Parlay Connector for authentication.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. Type a value in the **Authentication domain ID** field.
8. In the Additional Properties section, click **Gateway Authentication**.

9. Type a value in the **Parlay authentication type** field. This is the authentication type configured on the Parlay gateway for the client. It is passed to the Parlay gateway exactly as entered.
10. Type a value in the **Encryption preferences** field.
    - **none** if there is no encryption.
    - **des_56** for Data Encryption Standard (DES) with a 56-bit key.
    - **des_128** for Triple Data Encryption Standard (Triple DES) with a 168-bit key.
    - **triple_des** for Triple Data Encryption Standard (Triple DES) with a 168-bit key.
    - **rsa_512** for Rivest, Shamir, and Adleman (RSA) algorithm with 512-bit keys.
    - **rsa_1024** for Rivest, Shamir, and Adleman (RSA) algorithm with 1024-bit keys.
    - If you are using Parlay 4x authentication, type one of the following values in the **Authentication mechanism preferences** field:
      - P_OSA_MD5 - An MD5 hash
      - P_OSA_HMAC_MD5_96 - An HMAC-MD5 hash
      - P_OSA_HMAC_SHA1_96 - An HMAC-SHA1 hash
11. If Parlay 4x authentication is used, a value is required in the **Signing algorithms** field. Currently, `NONE` is the only Signing algorithm supported for 4x.
12. Type a value in the **Framework version** field to specify the framework version with which the client wishes to authenticate itself, so that it can work with the gateway services. Framework version is of the format P_PARLAY_4_2.
13. Optional: If you need a customized authenticator class for your target Parlay gateway, type a value in the **Authenticator classname** field. Type the full path and class name for the custom authenticator. If you use the default authentication implementation, you can leave this field blank.
14. Optional: If you need a customized authenticator callback class for your target Parlay gateway, type a value in the **Authenticator callback classname** field. Type the full path and class name for the custom authenticator. If you use the default authenticator callback implementation, you can leave this field blank.
15. Optional: If this configuration includes customized authenticator software that needs configuration data that is not included in the Parlay Connector configuration, click **Custom authenticator parameters** in the Additional Properties section.

    **Note:** The default authenticator implementation ignores any parameters that are specified here.
16. Optional: Complete these steps to define a new custom authenticator parameter.
    a. Click **New**.
    b. Type a value in the **Parameter name** field.
    c. Type a value in the **Parameter value** field.
    d. Click **Add**.
17. Optional: To change a custom authenticator parameter, type the **new value** in the appropriate row of the Parameter value column.
18. Click **OK**.
19. Optional: If this configuration includes customized authenticator callback software that needs configuration data that is not included in the Parlay Connector configuration, click **Custom authenticator callback parameters**.

Repeat the same steps used to define or change a custom authenticator parameter for the **Custom authenticator parameters** property, and then click **OK**.

**Note:** The default authenticator implementation ignores any parameters that are specified here.

#### Formatting a security certificate:

If your Parlay gateway provider specifies RSA encryption and provides a public key in the form of a security certificate, you might need to reformat a security certificate to configure this value.

#### Before you begin

The encryption certificate is provided in an ASCII file in this format.

```
-----BEGIN CERTIFICATE-----
MIIBIzCCAEegAwIBAwIBAAADBgEACCIxCzAJBgNVBAYTAlVLMRMwEQYDVQQDEwpB
ZXBvbmEgTHRkMB4XDTAyMTExMjE3NTE0M1oXDTAzMEExMjE3NNE0M1owIjEMMAkG
A6UEBhMCVUsxEzARBgNVBAMTCkFlcG9uYSBMdGQwgZ8wDQYJKoZIhvcNAQEBBQAD
nqR0AMIGJAoGBAJrAVSwEnJFExeW3Ug2kUPcZBzBP4A+IGWafDABk3Uc6KfcpQCvE
BejbFsHzffDL+QKYIkAs2WmDo1fjBjTqHNfsdpEDRAZuUlN6EwRpmBS6MujqKiJA
OCn6SkaDIBrQXte14JFZUX0ulZ4K8KY/NAb1tImGBRrQsi0pvbpip8PHAgMBAAEw
AwYBAAMBAA==
-----END CERTIFICATE-----
```

#### About this task

Complete these steps to convert the certificate to a single-line string.

1. Open the file with a **text editor**.
2. Ensure that word wrap is not enabled.
3. Add the new line character **(\n)** to the end of each line.
4. Add the **\n** to the end of the file. This is an example of the reformatted certificate. This is not an actual certificate.

   ```
   -----BEGIN CERTIFICATE-----\n
   MIIBIzCCAEegAwIBAwIBAAADBgEACCIxCzAJBgNVBAYTAlVLMRMwEQYDVQQDEwpB\n
   ZXBvbmEgTHRkMB4XDTAyMTExMjE3NTE0M1oXDTAzMEExMjE3NNE0M1owIjEMMAkG\n
   A6UEBhMCVUsxEzARBgNVBAMTCkFlcG9uYSBMdGQwgZ8wDQYJKoZIhvcNAQEBBQAD\n
   nqR0AMIGJAoGBAJrAVSwEnJFExeW3Ug2kUPcZBzBP4A+IGWafDABk3Uc6KfcpQCvE\n
   BejbFsHzffDL+QKYIkAs2WmDo1fjBjTqHNfsdpEDRAZuUlN6EwRpmBS6MujqKiJA\n
   OCn6SkaDIBrQXte14JFZUX0ulZ4K8KY/NAb1tImGBRrQsi0pvbpip8PHAgMBAAEw\n
   AwYBAAMBAA==\n
   -----END CERTIFICATE-----\n
   ```

5. Combine the lines into one string.
6. Copy this string into the **Security certificate configuration** field.

### Service agreements

A Service Level Agreement (SLA) is defined by the Parlay gateway and is used to verify that an application is authorized to use a service. The Parlay Connector signs service agreements on behalf of a telecommunications application.

The Parlay Connector must first select a service manager object and then obtain the object from the Parlay gateway before an application can use a service installed on the Parlay gateway. After the connector authenticates with the gateway, it determines what services are installed and active on the gateway and begins the process of selecting and obtaining service manager objects from the gateway. The connector must sign a service agreement with the gateway for each service manager it requests. The result of successfully signing a service agreement is a reference to the requested service instance.

The Parlay Connector uses configuration data to control dynamic loading of the service agreement signing classes. This enables you to accept the default service agreement signing classes or to configure customized service agreement signing classes.

### Service agreement processing

If the Parlay Connector computes or verifies a digital signature during the process of signing a service agreement, you must configure encryption. In most cases, this is the only configuration needed for the Parlay Connector to sign or terminate service agreements with the Parlay gateway.

The service agreement processing flow includes the following steps.
1. The Parlay Connector calls **initiateSignServiceAgreement** (serviceToken).
2. The Parlay gateway calls **signServiceAgreement** (serviceToken,agreementText, signingAlgorithm).
3. The Parlay Connector calls **signserviceAgreement** (serviceToken,agreementText, signingAlgorithm).

Typically, the client must use the same agreement text as the gateway. You do not need to configure agreement text unless the target gateway requires some other flow or does not use the same agreement text.

Typically, the client also uses the agreement text provided by the gateway as termination text. Therefore, you do not need to configure the Parlay Connector to terminate service agreements with the Parlay gateway. You might need to configure termination text if the target gateway requires termination text other than the agreement text provided by the gateway.

### Default service agreement classes

The default implementation of the ServiceAgreementManager interface begins the service agreement signing process and then waits for the Parlay gateway to call the ServiceAgreementCallback object. When the ServiceAgreementCallback object returns a digital signature to the gateway, the ServiceAgreementManager object requests a service manager object from the gateway. In this request, the ServiceAgreementManager object specifies the same agreement text and signing algorithm that the gateway specified when it called the ServiceAgreementCallback object.

The default implementation of the ServiceAgreementCallback interface computes a digital signature when called by the gateway to sign a service agreement. If the digital signature computed by the ServiceAgreementCallback object is accepted by the gateway, the gateway returns its own digital signature and the requested service manager object to the Parlay Connector.

### Custom service agreement

When customized processing is needed to obtain service manager objects from the Parlay gateway, the Service Agreement Manager class name and the Service Agreement Callback class name classes implement the required customized logic.

The class configured for Service Agreement Manager class name must implement the ServiceAgreementManager interface. This class provides the necessary NEP vendor-specific processing to sign service agreements with the Parlay gateway and returns the service manager object to the Parlay Connector.

The class configured for the Service Agreement Callback class name must implement both the ServiceAgreementCallback interface and the IpAppServiceAgreementManagement interface. This class responds to the Parlay gateway when called to sign service agreements.

Service agreement processing is usually symmetrical. This means you must customize the Parlay Connector to Parlay gateway processing and the Parlay gateway to Parlay Connector processing. Depending on the type of customization you might not need both customized classes.

The Parlay Connector creates a hash table containing its configuration and state information. This hash table is passed to the custom classes. This allows you to provide all configuration data needed by the custom classes during Parlay Connector configuration.

### Parlay gateway Service Agreement Management (signing algorithm preferences)

The Parlay gateway Service Agreement Management (signing algorithm preferences), initiates service agreements, passes agreement texts, required signing algorithms, and the return of digital signatures.

Signing a service agreement to obtain a Parlay gateway service manager includes three processes:

- The Parlay Connector initiates sign service agreement processing for the desired service manager. The Parlay Connector object that performs this step is the Service Agreement Manager.
- The Parlay gateway calls the Parlay Connector to sign a service agreement for the service manager, passing agreement text and the required signing algorithm, and verifying that the Parlay Connector returns the correct digital signature. The Parlay Connector object that performs this step is the Service Agreement Callback.
- The Parlay Connector calls the Parlay gateway to sign the service agreement and to return the service manager object, passing agreement text and the required signing algorithm. If the Parlay Connector returned the correct signature in the previous step, the Parlay gateway returns a digital signature and the service manager object. The Parlay Connector object that performs this step is the Service Agreement Manager. A customized plug-in class can override either or both of the Service Agreement Manager and Service Agreement Callback.

### Default Parlay gateway Service Agreement Management

The default implementation of the Service Agreement Manager interface begins the service agreement signing process and then waits for the Parlay gateway to call the Service Agreement Callback object.

The Service Agreement Callback object will return a digital signature to the Parlay gateway, while the Service Agreement Manager object requests a service manager object from the Parlay gateway. In this request, the Service Agreement Manager specifies the same agreement text and signing algorithm that the Parlay gateway specified when it called the Service Agreement Callback object.

If the Parlay gateway accepts the digital signature computed by the Service Agreement Callback object, then the Parlay gateway returns its own digital signature and the requested service manager object to the Service Agreement Manager. The default implementation of the Service Agreement Callback interface computes a digital signature when called by the Parlay gateway to sign a service agreement.

### Managing the service agreement for a Parlay gateway

If custom support is needed for to manage the service agreement with a Parlay gateway, either the Service Agreement Manager class name configuration field or the Service Agreement Callback class name configuration field, or both, can be used to specify custom classes.

The class specified in the Service Agreement Manager class name configuration field must implement the ServiceAgreementManager interface. The class specified in the Service Agreement Callback class name configuration field must implement the following interfaces:

- com.ibm.wast.parlay.connector.ServiceAgreementManager
- com.ibm.wast.parlay.api.fw.fw_application.service_agreement.IpAppServiceAgreementManagement

### Service selection

When there is more than one service manager instance for a service type and each instance has different properties, the Parlay Connector might not be able to determine which service manager instance to use for signing service agreements with the Parlay gateway. If all of the service manager instances for a service type meet the needs of your application, you do not need to specify service properties for a service type. Otherwise, you might need to define service properties to ensure that the Parlay Connector selects the correct service manager instance from the Parlay gateway.

### Local service managers

Although Parlay service managers are normally on remote computers, you can configure to obtain service managers implemented locally if the service managers meet certain requirements.

You might use a local service manager implementation for these reasons.

- Your Parlay gateway does not support a service type needed for your application.
- A service type on your Parlay gateway does not meet the performance requirements of your application.
- Your application needs to integrate with a specific Service Provider Network (SPN) element that your Parlay gateway does not support.

If you use a local service manager implementation, your application can use the Parlay Connector getServiceManager() API to obtain service manager instances only if the implementation meets these requirements.

- The implementation must be a stateless EJB that implements these interfaces:

IpService
IpServiceProperties

- The service type name and the JNDI name of the implementation must be defined in the Parlay Connector configuration.
- The JNDI name of the implementation must be bound to the implementation in the ParlayConnectorEJB deployment descriptor.

You can configure the Parlay Connector with a local service manager implementation using a service type name defined on the Parlay gateway. In this case, when your application specifies that service type name in a Parlay Connector API, the Parlay Connector considers the specified service type to be the local implementation and not the service type that the Parlay gateway provides.

**Service types:**

The Parlay specification defines a set of service types. Each service type has a service type name and an interface definition. Each service type also has a set of properties that define its capabilities.

When there is more than one service instance of a service type, the instances usually have different properties. If all of the service manager instances meet the needs of your application, you do not need to specify service properties for a service type. Otherwise, you must define service properties to ensure that the Parlay Connector selects the correct service manager from the Parlay gateway.

The Parlay Specification defines a set of service type names as possible values for TpServiceTypeName. Two gateways might define additional service type names. One gateway might define the names as additions to the specified list. Another gateway might define the names as replacements to the names in the list. In either case, both gateways are compliant with the Parlay specification.

Because service type names might vary from one gateway to another, applications must allow for configurable service type names. Before requesting a service manager, the application must locate the exact service type name in its configuration data.

**Service managers**

The Parlay Connector uses the default behavior to obtain a service manager from the Parlay gateway. By default, the connector specifies a service type and requests a single service manager from the gateway. If there is more than one service manager for the service type registered on the gateway, the gateway uses its own criteria to determine which service manager to return to the connector. You can control which service manager is returned to the connector by defining environment variables.

Although service managers are normally on remote computers, you can configure to obtain service managers implemented locally if the service managers meet certain requirements.

A Parlay client application obtains Parlay service managers by requesting them from the Parlay Connector as specified in the Parlay Connector APIs. The service managers available to a Parlay client application depend on the Parlay gateway and the service agreements in effect for the Parlay client. The vendor provides a list of Parlay services and the properties for each.

### Multiple services of the same type

When you define service types for the Parlay Connector, you must create a unique definition for each service type. Therefore, you might have service types with the same name that have different properties. To differentiate between service types that have the same name but different properties, you can assign an alias name to represent a service type with specific properties.

For more information, refer to the topic *Assigning aliases to service types*.

### Nonstandard service type names

As part of configuring Parlay Connector initialization, you can configure for nonstandard gateway services. This includes specifying nonstandard service type implementations.

A Parlay gateway might provide more than one service manager for a service type, each with its own set of properties. In this case, the Parlay gateway might use the standard service type name for one of the service managers and define a nonstandard service type name for each of the other service managers.

To configure the Parlay Connector for a gateway service that uses a nonstandard service type name with a standard interface definition, you must have the following information:
- The service type name used on the gateway
- The standard service type name for the service

### Assigning aliases to service types:

When you define service types for the Parlay Connector, you must create a unique definition for each service type. Therefore, you might have service types with the same name that have different properties. To differentiate between service types that have the same name but different properties, you can assign an alias name to represent a service type with specific properties.

### About this task

Every service manager name has two parts, for example *ServiceType*:*ServiceID*. The combination is unique for the Parlay gateway, and therefore for the Parlay Connector. When you assign an alias for a service manager, your applications can then use that alias to obtain access to the service manager.

If you have a service type called connection type, and its properties are cable, dial up, and DSL, you might define three aliases for the one service type, but each with a different connection property definition. When you want to use the cable connection, you must specify the cable alias. By using an alias for each, you are guaranteed to get the connection type you requested.

You might also have service types with different names but the same properties. An array of service types can be defined for each connection type, such as multiple service types which use the cable connection type. In this case, all the service type names are treated as aliases. A dummy alias with the same name as the service type is automatically created, so each alias is a list of services from *1* to *n* entries.

A record of which service type was last used is kept so that the next request for that service type uses the next entry in the list. This feature spreads the demand across the services, balancing the load.

A service type explicitly defined under an alias is also defined under the implicit alias, which is the service name. So if alias A is defined for a service type X with a specific property definition of X, all service types in alias A are in service type X, but not all service types in X are in alias A.

In this case, If a user application needs a service type specifically defined by alias A, and another application needs the same service type, if the properties of service type X are irrelevant for both applications, the applications might receive the same service type. Load balancing is not an issue because the service type is specified from two different lists, and load balancing only knows about the service types in the alias list.

Complete these steps to assign an alias for a service type.

1. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
    b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click **Telecom Applications** → **Parlay Connector** → **Parlay Connector Template** → **Svc Type Alias Defs**.
5. Assign aliases for one or more service types.
6. Click **OK**.

**Configuring service selection:**

Configure service selection for the Parlay Connector if you need to specify properties for a service type. You might need to specify properties for a service type if the Parlay Connector needs to verify which service manager instance to use for signing service agreements with the Parlay gateway.

**About this task**

Complete these steps to configure the Parlay Connector for service selection.

1. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

*host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

*port* is the secured port used to access the console. The default port is *9043*.

**Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Service Manager - Properties**.
8. Complete these steps to define a new service type.
   a. Click **New**.
   b. Type a value in the **Service Type Name** field.
   c. Type a value in the **Property Name** field.
   d. Click **Add**.
9. In the **Service property value list** column, type a value in the text field for each property that requires a value to create a service property values list for the service manager property.
10. Click **OK** to save your changes.

**Processing service level agreements:**

When you configure the Parlay Connector to obtain service managers from the Parlay gateway, you must also configure the connector to process service level agreements.

**About this task**

Complete these steps to configure the Parlay Connector for processing service level agreements.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)

   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Service Agreements**.
8. Optional: Type a value in the **Service agreement - signing retries** field. The default value is `4`.
9. Optional: Type a value in the **Service agreement signing retry delay** field in seconds .
10. Optional: Type a value in the **Service agreement - manager classname** field. The default value is `none`. If you use the default service agreement manager implementation, you do not need to type a value in this field.
11. Optional: Type a value in the **Service agreement - callback classname** field. The default value is `none`. If you use the default service agreement manager implementation, you do not need to type a value in this field.
12. Optional: In the Additional Properties section, click **Service agreement manager parameters** if this configuration includes customized service agreement manager software that needs configuration data not included in the Parlay Connector configuration.
13. Optional: Complete these steps to define a new custom parameter. The default service agreement manager implementation ignores parameters defined here.
    a. Click **New**.
    b. Type a value in the **Parameter name** field.
    c. Type a value in the **Parameter value** field.
    d. Click **Add**.
14. Optional: To change a custom parameter, type the new value in the appropriate row of the Parameter value column.
15. Click **OK**.
16. Optional: Click **Service agreement callback parameters** if this configuration includes customized service agreement callback software that needs configuration data not included in the Parlay Connector configuration. Repeat the same steps used to define or change a custom parameter for the **Service agreement manager parameters** property, and then click **OK**.

**Specifying agreement text:**

The Parlay Connector requests a service manager from the Parlay gateway. This action initiates a service agreement handshake. The agreement text proposed by the Parlay gateway is used. You do not have the option of configuring your own agreement text.

**Configuring termination text:**

When the Parlay Connector terminates a service level agreement that it has signed with the Parlay gateway, the connector must specify termination text to the gateway. Configure termination text for the Parlay Connector if the Parlay gateway requires client applications to use a text that is not the same as the Agreement text. Configuration can be specific to a service type or to the host, or it can be a default value.

**About this task**

Complete these steps to configure the Parlay Connector for termination text.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Service Agreements**.
8. In the Additional Properties section, click **SLA termination text**.
9. Select a value from the **Source of termination text** list. The default value is `use_AgreementText_for_TerminationText`.
   - **use_AgreementText_for_TerminationText**.
   - **TerminationText_configured_in_Connector** .
10. Type a value in the **Termination text for host** field.
11. In the Additional Properties section, click **Termination text by service manager**.
12. Optional: Complete these steps if you need to define termination text for a service type not already listed.
    a. Click **New**.
    b. Type a value in the **Service type or alias** field.
    c. Type a value in the **Termination text** field.
    d. Click **Add**.
13. Optional: To change the termination text for a service type already listed, type the new value in the appropriate row of the Termination text column.
14. Click **OK**.

**Configuring local service managers:**

You might need to configure your system to obtain a local service manager if your Parlay gateway does not support a service type needed for your application. The

Parlay gateway might not support a service type if the service type on your Parlay gateway does not meet the performance requirements of the application, or if the application needs to integrate with a Service Provider Network (SPN) element that the Parlay gateway does not support.

**Before you begin**

A local service manager used as your Parlay gateway must meet these requirements to use the Parlay Connector getServiceManager() API to obtain service manager instances.

The implementation must be a stateless Enterprise Java Bean (EJB) that implements both the IpService and the IpServiceProperties interfaces.

The service type name and the JNDI name of the local service manager must be defined in the Parlay Connector configuration.

The JNDI name of the local service manager must be bound to the ParlayConnectorEJB deployment descriptor.

**About this task**

Complete these steps to configure the Parlay Connector for local service managers.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, select **Service Manager - Local Implementations** .
8. Optional: Complete these steps to define a new local service manager implementation.
   a. Click **New**
   b. Type a value in the **Service Type Name** field.
   c. Type a value in the **Implementation JNDI name** field.
   d. Click **Add**.

9. Optional: To change an existing local service manager implementation, type the new value in the appropriate row of the Implementation JNDI name column.
10. Click **OK** to save your changes.

## Generic event notifications

To help it react to a failover recovery of the service manager, the Parlay Connector can use two different interfaces to receive events. These events notify the Parlay Connector when the service manager goes offline and when it becomes available again.

### Interface: IpAppEventNotification

The Parlay Connector uses the IpAppEventNotification interface to receive events related to the availability and unavailability of the service manager.

The notification registration for IpAppEventNotification occurs after the Parlay Connector has signed service agreements with the Parlay gateway. The Parlay Connector queries for the list of service types available at the Gateway and registers event notifications for each of the service types.

The implementation of this interface applies to both 3x and 4x Parlay gateways.

**Gateway callback method: reportNotification()**

Applies to: Parlay 3x and 4x

Events reported: Service availability and unavailability

### Interface: IpAppFaultManager

The Parlay Connector also implements the IpAppFaultManager interface to receive events related to the availability and unavailability of the service manager.

Different methods are used for Parlay 3x and Parlay 4x.

**Gateway callback method: svcUnavailableInd()**

Applies to: Parlay 3x

Events reported: Service unavailability only

**Gateway callback method: svcAvailStatusInd()**

Applies to: Parlay 4x

Events reported: Service availability and unavailability

Note that availability cannot be reported in Parlay 3x through the IpAppFaultManager interface.

## Gateway

The characteristics of the Parlay gateway determine some application processing.

The Parlay Connector has application programming interfaces (APIs) that telecommunications applications can use to obtain Parlay gateway service managers. The connector APIs also provide information relevant to the processing done by telecommunications applications.

Some gateway vendors interpret the Parlay specification differently. These differences can affect application processing flow. You can increase the portability of your application by using configuration data to define the requirements for your gateway.

You can configure the gateway characteristics to the connector, and the applications can obtain the information at run time using the connector APIs. When you configure the gateway characteristics to the connector, you configure them once for all of the applications instead of configuring them for each application.

### Configuring notifications:

When differences in gateways affect processing flow, you can configure data to define the requirements for your Parlay gateway. Configure notifications for properties that vary for different Parlay gateways.

#### About this task

Complete these steps to configure the Parlay Connector with service notification information.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Notifications**.
8. Optional: Select a value from the **Reset service notifications** drop-down list. The default value is `true`.
   - **true** if the Parlay Connector should cancel outstanding notifications during the start or restart process.
   - **false** if the Parlay Connector should not cancel outstanding notifications during the start or restart process.
9. Optional: Select a value from the **GenericCallControl - event location** drop-down list. The default value is `terminating`.
   - **terminating** to request event notification for the terminating switch.
   - **originating** to request event notification for the originating switch.

10. Optional: Select a value from the **GenericCallControl - event type** drop-down list. The default value is `analysed`.
    - **analysed** to request Event notification for the address-analysed event.
    - **collected** to request event notification for the address-collected event.
11. Optional: Select a value from the **MultiPartyCallControl - event type** drop-down list. The default value is `analysed`.
    - **analysed** to request event notification for the address-analysed event.
    - **collected** to request event notification for the address-collected event.
12. Click **OK** to save your changes.

**Configuring custom properties:**

When you configure the Parlay Connector runtime environment, you can configure the MultiPartyCallControl (MPCC) custom properties.

**About this task**

**Note:** This section illustrates how custom properties can be configured for MPCC or GCC as well.

Complete these steps to configure the MPCC custom properties.

1. Log in to the Integrated Solutions Console:
    1. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
    2. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    3. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Notifications**.
8. In the Additional Properties section, click **MultiPartyCallControl - custom properties**.
9. Optional: Complete these steps to define a new custom property.
    1. Click **New**
    2. Type a value in the **Property name** field.
    3. Type a value in the **Property value** field.
    4. Click **Add**.

10. Optional: To change the value of a custom property, type the new value in the appropriate row of the Property value column.
11. Click **OK**.

**Configuring load balancing:**

You can configure load-balancing to define the criteria for processing load information. These criteria are used by the Parlay Connector when responding to a Parlay gateway request.

**About this task**

Complete these steps to configure the Parlay Connector for load balancing.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Parlay Connector**.
6. Click the name of an active configuration.
7. In the Additional Properties section, click **Load Balancing**.
8. Type a decimal number in the **Load percentage** field, such as `.5`. The default value is `0.4`.
9. Type a decimal number in the**Overload percentage** field, such as `.75`. The default value is `0.6`.
10. Select a value from the **Load qualifier** drop-down list.
    - **any** to report an overload if any clone in the cluster is overloaded.
    - **all** to report an overload if all clones in the cluster are overloaded.
11. Click **OK** to save your changes.

**Custom gateway configuration:**

The key features of the Parlay gateway that you might need to customize include the gateway location methods, the framework authentication methods, and the service agreement signing methods.

The Parlay Connector provides several ways to customize the Parlay gateway connectivity. These sections provide general information about customizing the gateway.

**Custom gateway locator**

If the Parlay gateway uses a method other than the default locator class to provide its initial access object (IpInitial) to clients, customized locator software is needed for the gateway location server to provide initial contact information to the Parlay Connector.

When customized processing is needed to locate the gateway, you can configure a class that implements the required customized logic. The customized gateway locator class provides the necessary network equipment provider (NEP) vendor-specific processing for the connector to obtain initial access to the gateway.

The Parlay Connector creates a hash table containing its configuration and state information. This hash table is passed to the custom class. This lets you provide all configuration data needed by the custom class during Parlay Connector configuration.

**Custom authentication**

The authentication process uses the standard Parlay APIs as defined by the IDL and the Parlay Group. The Parlay specification defines the authentication process, though some aspects of the process can be implemented in different ways. You can configure customized authentication classes if the default implementation cannot be configured to authenticate with your Parlay gateway.

When extensive customized processing is needed to authenticate with the Parlay gateway, the Authenticator class name and Authenticator callback class name configuration fields specify the classes in the class path that implement the required customized logic.

The Parlay Connector creates a hash table containing its configuration and state information. This hash table is passed to the custom classes, allowing you to provide all configuration data needed by the custom classes during Parlay Connector configuration.

**Custom gateway properties**

The Parlay Connector lets you customize initialization. If the custom software used for your target Parlay gateway needs configuration data not already included in the Parlay Connector configuration, you can add properties to the Parlay Connector configuration so that the values you assign to them will be passed to the custom software.

When the Parlay Connector determines that custom software is being used, it creates a hash table containing all of its configuration information and some of its state information. The Parlay Connector then obtains and initializes an instance of the custom class and passes this hash table to the instance.

## Configuring the Application Manager

The Application Manager manages Parlay applications. You can configure the Application Manager to change the way in which it manages your Parlay applications.

### Specifying configuration parameters:

To interact with the Application Manager, you will normally use the Parlay Administration Console. However, you can also configure the Application Manager itself to determine how it will manage your Parlay applications.

#### Before you begin

You must ensure that the WebSphere Application Server has started.

#### About this task

Certain environment variables must be enabled in the deployment descriptor of the Application Manager. Although the variables are set automatically and do not normally need to be changed, you can examine them and change their settings. For example, you might decide to change some settings for performance tuning.

Software that is customized for a particular gateway might require additional environment variables. The environment variables are required at startup, and they might require redeployment of the application.

**Note:** The components and applications managed by the Application Manager are called *managed objects*.

To configure the Application Manager, complete these steps.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, expand **TWSS Parlay Administration**.
3. Click **Telecom Server** to display the list of deployed telecom applications.
4. Click the name of the application you want to configure, for example **Parlay Connector**.
5. Click **Application Manager**.
6. Select configuration values for the Application Manager as follows:

   **Auto-start Enabled**
   If `true`, the Application Manager starts itself automatically when WebSphere Application Server starts. If `false`, the **Start** button is the only way to start the Application Manager. The default value is `false`.

   **Cluster initialization sleep time in seconds**
   During startup, the number of seconds to wait for the cluster to come up before setting the appropriate Application Manager state. This value

is used for synchronizing the startup sequence of the nodes in the cluster. The default value is `10` seconds.

**Parlay Event Queue - max threads**
The number of QueuedObjectEvent threads that need to run. The default value is `8`.

**Parlay Event Queue - dispatch mode**
The algorithm mode used by the Application Manager to dispatch events. Select one of the following values:

*****
Round-robin: events are dispatched across all computers in the cluster.

**1**
Local: events are dispatched locally.

**i**
Idle: events are dispatched based on workload.

7. Optional: Specify values so that the Application Manager can write events using the Fault and Alarm component Web service, as follows:

   **EndPoint URI**
   The endpoint URI for the Fault and Alarm component Web service, for example http://localhost:9080/TWSS/ServicePlatform/FaultAlarm/services/FaultAlarm

   **Username**
   The user name for application-level authentication

   **Password**
   The password for application-level authentication

   **Timeout**
   The request timeout, in seconds, for invoking the Fault and Alarm component Web service
8. Click **OK** to save your changes.
9. Optional: Restart the application for your changes to take effect if you configured either or both of the following values:
   - Auto-start enabled
   - Cluster initialization sleep time in seconds

**What to do next**

To configure additional applications, or managed objects, to be managed by the Application Manager, follow the steps in the topic *Specifying Parlay applications to be managed*.

**Configuring Parlay applications to behave as managed objects:**

The components and applications managed by the Application Manager are called *managed objects*. Every application that needs management events from the Application Manager must define a stateless bean implementing the ManagedObjectOperations interface. The bean must be deployed with a predefined JNDI name format. When this is done, the application becomes a managed object.

The Application Manager maintains the list of managed objects–that is, the list of Parlay applications that implement the ManagedObjectOperations interface–to which it will send the management events.

In previous versions of Telecom Web Services Server (through version 6.2), a list of EJB references was attached to the Application Manager deployment descriptor to provide EJB references to these managed object EJBs.

In Telecom Web Services Server version 7.0, the Application Manager looks for managed objects using a JNDI tree that has a specific namespace. The JNDI namespace for each managed object is in the following format:`ejb/WAST/Application/`*`sequence_number`*`/`*`parlay_connector_name`*`/`*`application_name`*`/`*`remote_interface_name`*

where

*sequence_number* is an integer that determines the order in which this managed object appears in the Telecom Applications list in the Parlay Administration Console

*parlay_connector_name* is the name of the Parlay Connector instance that will manage this managed object

*application_name* is the name of the managed Parlay application

*remote_interface_name* is the name of the remote interface implemented by the managed object

When a Parlay application defines a stateless EJB that is a managed object, it will define the JNDI name of the EJB using this format. The JNDI name should include the name of the Parlay Connector, which can be unique for each Parlay Connector instance. This enables the Application Manager to find all of the stateless EJBs that are managed objects for a given instance of the Parlay Connector.

The list of managed objects is displayed in the Parlay Administration Console when the Parlay Connector application EAR file has been started. It is always preferable to start the Parlay application EAR files before starting the Parlay Connector application EAR file. That way, the managed object registrations are complete and can be detected by the Application Manager.

## Configuring naming schemes

You can configure naming schemes for groups and uniform resource indicators (URIs).

### Group schema configuration

The user or administrator can use the TWSS Administration Console to specify the list of schemas to support. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.

#### About this task

The user or the administrator who is specifying the supported group schema, should be aware of the what groups he or she wants to support. The groups are internally defined by the IBM XDMS server.

This configuration applies to the following Web service implementations:

- WAP Push over SMPP
- Parlay X SMS over SMPP
- Parlay X Terminal Location over MLP
- Parlay X Terminal Status over Parlay

- Parlay X Terminal Location over Parlay
- Parlay X SMS over Parlay

1. Open the TWSS Administration Console.
2. Navigate to the Common Service Settings page and open the **Runtime** tab.
3. Populate the General Properties fields. Depending on which Web service implementations you are working with, the fields will include a subset of the following:
   - **Country and Code**: The local country and code. The default setting is `United States [1]`.
   - **Address Plan**: The local address plan. The default value in the drop-down list is `com.ibm.mds.comm.support.E164AddressPlan`.
   - **Enable Group Resolution**: Whether or not to resolve the group URI to member URIs. The default value in the drop-down list is `true`.
   - **Periodic Notification Supported**: Whether or not periodic notification is enabled. The default value in the drop-down list is `true`.
   - **Geographical Notification Supported**: Whether or not notifications may be set on a geographical area. The default value in the drop-down list is `true`.
   - **Terminal Location Enabled**: Whether or not the Terminal Location Web service is enabled. The default value in the drop-down list is `true`.
   - **Network Resource Name**: The default value is `PX21_TL_MLP`.
   - **Enable Transaction Monitoring**: The default value in the drop-down list is `false`.

     **Note:** If you specify `true`, then all transactions are monitored. A warning is issued when a transaction remains in the same state for a time period exceeding that specified in the service policy service.common.Monitor.stateTransition.warning. A warning and an alarm are issued when the time period in the service policy service.common.Monitor.stateTransition.alarm is exceeded.
   - **Supported Group Scheme**: The group scheme to use. The default value is `glmgroup`.
4. Enable the Service Platform components you want by inserting their endpoint URIs in the Common Components section.
5. Click **OK**. The group schema is defined within the TWSS Administration Console.

## Results

If an user sends an SMS to a group that is defined in IBM XDMS, a group schema check is made with the group that is defined in the administration console. If the IBM XDMS and the group schema defined in the Administration Console match each other, then the group will be processed.

These settings are not used in ESB. This attribute does not configure the AG schema, and the IBM XDMS allows the user to configure many other group schema's apart from **group**. The attribute configuration indicates the service implementation concerning the group schema's that the SI wants to support. If the group schema passed from the IBM XDMS is present in the supported group schema attribute, the group will be processed and the requested operation is performed on each target.

**Note:** The user can define a list of group schemes separated by a comma (,), which acts as a delimiter.

Here is an example of group schemes: **mygroup, glmgroup, ibmgroup**. The default supported group scheme name is **glmgroup**.

### Configuring a URI scheme

URIs and URI references are organized with a top-level naming structure which follows a scheme or design name.

A URI is defined by one of many standard schemes. Refer to IETF RFC 2396 for details about semantics and syntax.

This is an example of a URI scheme:

```
<scheme>:<usersinformation>@<thehost>:<port><path>;<parameters>
```

URI schemes or protocols often have the exact same name, and they can be used for a variety of purposes. Most commonly, the Hypertext Transfer Protocol (http) scheme is used for Web interaction. URIs are also used for XML namespaces and RDF resource identifiers, which are unrelated to the protocols. Finally, there are instances in which the scheme is not connected to a protocol at all (for example, a simple directory path such as `file`).

## Configuring privacy integration

The Telecom Web Services Server product does not provide a privacy common component implementation. However, if you have a privacy service deployed in your network, the Web service implementations are configured by default to integrate with it.

### About this task

Additionally the Telecom Web Services Server product is configured so that the privacy common component interface integrates with a privacy service and is enabled by default. This ensures that the Web service implementations are secure when first deployed.

All Web service implementations come with the client-side portion of the privacy Web service, so that each Web service implementation can integrate with your environment. If a privacy service is not available, you should disable the configuration for the interface to the privacy Web service.

If a privacy service does not exist in the network, use the Telecom Web Services Server Administration Console to disable the configuration for the privacy component. To disable all Service Platform components, the endpoint URI should be blank.

To enable or disable the privacy client, complete the following steps.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

*port* is the secured port used to access the console. The default port is *9043*.

**Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services**.
4. Click the appropriate Web service, such as **IMS Call Notification**.
5. Click **Privacy Client**.
   - To disable the privacy client, the **End Point URI**, **User Name**, and **Password** should be blank.
   - To enable the privacy client, specify the **End Point URI**, **User Name**, and **Password**

## Configuring communication with the SMSC

The SMPP-based Web service implementations (SMS over SMPP and WAP Push over SMPP), use predefined values when communicating with the Short Message Service Center (SMSC). The values are used in each message sent to the SMSC. You can modify these values using the TWSS Administration Console.

For additional details about configuring the SMPP-based Web service implementations, refer to their respective *Administering* topics.

### Preparation

Before configuring for communication with the SMSC, you must have completed the following tasks:

- Installed and configured WebSphere Application Server
- Installed the WebSphere Telecom Web Services Server base components, including the Telecom Web Services Server Administration Console
- Installed one or both of the SMPP-based Web service implementations: SMS over SMPP and WAP Push over SMPP
- Installed and configured the JCA adapter (this is normally done using the First Steps script)

### List of ESMC settings

ESMC settings represent the SMSC default values that will be used in each message sent to the SMSC.

If you want to use settings that are different from the SMPP default values, open the TWSS Administration Console and navigate to the Message Data Services Configuration section. Click **SMS SMPP ESMC Settings** and set the following properties on the **General Property Runtime** tab:

**ESMC SOURCE TON**
Type of Number (TON) to be used in the Short Messaging Entity (SME) address parameters for the message source. The default value is 1 (international).

**ESMC SOURCE NPI**
Numeric Plan Indicator (NPI) to be used in the SME address parameters for the message source. The default value is 1 (ISDN E163/E164).

**ESMC DESTINATION TON**
TON to be used in the SME address parameters for the message destination. The default value is 0.

**ESMC DESTINATION NPI**
NPI to be used in the SME address parameters for the message destination. The default value is 0.

**ESMC ESM CLASS**
Special message attributes associated with the short message. The default value is 0 (none).

**ESMC PROTOCOL ID**
Protocol ID used to route the messages. The default value is 0 (none).

**ESMC PRIORITY FLAG**
Allows the originating SME to assign a priority level to the short message. The default value is 0.

**ESMC REPLACE IF PRESENT FLAG**
Used to request that the SMSC replace a previously submitted message that is still pending for delivery. The field is a drop-down list containing the following choices:

Don’t replace [0]

Replace [1]

The default value is 1 (replace).

**ESMC MESSAGE ID**
An integer in the range 0–254 that specifies the SMSC index for a predefined or ″canned″ message. The default value is 0. This indicates a reserved value.

**ESMC ESM CLASS SAR**
Fragmentation using SAR. The default value is 0.

**Message ID Format**
Format for message IDs returned by the SMSC in submit_sm_resp and submit_multi_resp protocol data units (PDUs). This is an opaque value and must set according to the SMSC implementations. The field is a drop-down list containing two choices:

**HEXADECIMAL**: The returned message ID format will be hexadecimal.

**DECIMAL**: The returned message ID format will be decimal. This is the default value.

## Implementation notes

When you change these values, it is not necessary to restart your system for the changes to take effect.

An SMSC delivery receipt is requested when the `ConfirmedDelivery` flag is set to `true`, either in the message itself or in the TWSS Administration Console (connection properties for the backend server).

When you are deploying Parlay X SMS over SMPP or WAP Push over SMPP, it may be necessary to update the configuration for the SMSC for your network.

Consequently, you need to understand the SMSC behavior before you adjust the ESMC settings in the TWSS Administration Console.

## Enabling the Common Event Infrastructure Service

The Fault and Alarm component Web service can be configured to use the Common Event Infrastructure (CEI) service that is included with WebSphere Application Server and WebSphere Enterprise Service Bus. You must enable the service for the Access Gateway cluster and for the cluster on which the Web service implementations are deployed.

### Before you begin

The instructions for this procedure assumes the database server is not on the same server as the WebSphere Application Server.

You must have:

- Installed the database client application on the WebSphere Application Server node.
- Stopped the deployment manager and nodes for the Access Gateway cluster
- Stopped the deployment manager and nodes for the Web services cluster

### About this task

By default the CEI service is enabled and configured to use the database that is included with WebSphere Application Server. For a production environment, configure the CEI service to use IBM DB2 Enterprise Server Edition or Oracle.

To configure the CEI service database, you must run a *wsadmin* command. The command is provided in the instructions. The command is based on the following configuration. If the configuration does not match your environment, consult the WebSphere Application Server Information Center for details. Refer to the related reference link at the end of the instructions.

This procedure assumes:

- A cluster network configuration
- The database server in not on the same server WebSphere Application Server
- The database server is running and configured
- The database client application is installed on the WebSphere Application Server node
- **Oracle** The database that will store the CEI events has been created

1. On the Access Gateway deployment manager, configure the CEI repository. Refer to the topic *Configuring the event database* in theWebSphere ESB Information Center for more information. (A link is provided at the end of these instructions.) Select the appropriate database setup for your network.
2. Stop the manager on the network deployment server for the service implementations cluster.

   **Note:** The user name and password *stopmanager.sh* are required only when the security is enabled.

   Run the following command:

   AIX
   ```
   was_profile_root/bin/stopManager.sh -username user_name
   -password password
   ```

Linux

```
was_profile_root/bin/stopManager.sh -username user_name
-password password
```

```
was_profile_root/bin/stopManager.sh -username user_name -password
password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

3. Run the deployEventService command to enable CEI.
   a. Change to the deployment manager profile directory. For example, if you are using the default path:

      ```
      cd /opt/IBM/WebSphere/AppServer/profiles/Dmgr01/bin
      ```

   b. Run the following command:

      ```
      ./wsadmin.sh -conntype none -c "\$AdminTask deployEventService { -clusterName cluster_name -enable true }"
      ```

      Where:

      *cluster_name* is the cluster where the Web service implementations are deployed.
4. Start the deployment manager and nodes. For example, from the `/bin` directory of the deployment manager profile:

   ```
   ./startManager.sh
   ```

5. Click **Security** → **Global security** and enable both administrative security and application security.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Security** → **Secure administration, applications, and infrastructure**.
6. Verify that the service was enabled:
   a. In the navigation pane, click **Servers** → **Server Types** → **WebSphere application servers**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Application Servers**.
   b. Click the name of the application server.
   c. Under Container Settings, click **Container Services** and then verify that **Common Event Infrastructure Service** is visible.
7. To enable the service on startup, click **Common Event Infrastructure Service**.
   a. On the configuration page, click **Enable service at server startup** option.
   b. Click **Apply**.
8. Stop the manager before configuring the CEI service repository. Run the following command:

   AIX

   ```
   was_profile_root/bin/stopManager.sh -username user_name
   -password password
   ```

   Linux

   ```
   was_profile_root/bin/stopManager.sh -username user_name
   -password password
   ```

```
was_profile_root/bin/stopManager.sh -username user_name -password
password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

9. Configure the CEI repository by running the appropriate command for your database server:
   - **DB2** Run the following command. Note that this command is entered on a single line.

     ```
     ./wsadmin -conntype none -c "\$AdminTask configEventServiceDB2DB {-createDB true
     -overrideDataSource true -clusterName cluster_name
     -jdbcClassPath path_to_db_jar -dbHostName host_name
     -dbUser db_user -dbPassword password -dbPort port}"
     ```

     Where:

     `-createDB true` indicates that the database has not been created, but the database server is running. If the database has been created, specify `false` and refer to the WebSphere Application Server Information Center for more information.

     *cluster_name* is the cluster where WebSphere Telecom Web Services Server will be deployed and where the data source for the CEI service should be created.

     *path_to_db_jar* is the path to the database JAR file.

     *host_name* is the server on which the database server is deployed.

     *db_user* is the database administrator ID, for example db2inst1.

     *password* is the password for the database administrator ID.

     *port* is the port used to communicate with the database server. The default value is 50000.
   - **Oracle** Run the following command. Note that this command is entered on a single line.

     ```
     ./wsadmin -conntype none -c "\$AdminTask configEventServiceOracleDB {-createDB true -override
     ```

     Where:

     `-createDB true` indicates that the database is configured and running. If the database has not been created, specify `false` and refer to the WebSphere Application Server Information Center for more information.

     *cluster_name* is the cluster where WebSphere Telecom Web Services Server will be deployed and where the data source for the CEI service should be created.

     *path_to_db_jar* is the path to the database JAR file.

     *oracle_home* is the directory where Oracle is installed.

     *db_user* is the Oracle schema user ID that will own the CEI tables. The user ID will be created during the database creation. The default value is `ceiuser`.

*cei_password* is the password of the schema user ID. The password will be created when the database is created. This parameter is required.

*db_name* is the Oracle System Identifier (SID). The SID must already be created and available for the command to create the tables and populate the tables with data.

*host_name* is the server on which the database server is deployed

*port* is the port used to communicate with the Oracle instance. The default value is 1521.

*dba* is the database administrator ID with SYSDBA privileges. The default value is `sys`.

*dba_password* is the password for the database administrator ID.

10. Start the manager. Run the following command:

    AIX *`was_profile_root`*`/bin/startManager.sh`

    Linux *`was_profile_root`*`/bin/startManager.sh`

    *`was_profile_root`*`/bin/startManager.sh`

    Where:

    The *`was_profile_root`* path contains the name of the deployment manager profile (for example, Dmgr01).

11. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
    b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    c. Click **Log in**.
12. In the navigation pane, click **Resources** → **JDBC** → **JDBC Providers**.
13. Click the JDBC provider for your database:

    DB2 Event_DB2_JDBC_Provider

    Oracle Oracle JDBC Driver
14. Under Additional Properties, click **Data sources**.
15. Verify that following names are listed in the JNDI name column.

    jdbc/cei

    jdbc/event_catalog
16. DB2 For CEI to work on your DB2 server, configure the following JNDI: `jdbc/com.ibm.ws.sib/imsCluster-CommonEventInfrastructure_Bus`.
17. In the navigation pane, click **Resources** → **JDBC** → **JDBC Providers**.
18. Click **New** and enter the following information for the newly created DB2 Universal JDBC Driver Provider:

    Populate :

   a. Enter the JNDI name as `jdbc/com.ibm.ws.sib/imsCluster-CommonEventInfrastructure_Bus`.
   b. Select a proper J2C for the Component-managed authentication alias.
   c. Enter `EVENT` as the database name.
   d. Enter your database server name as the Server name.
   e. Enter the port number for the database server, for example 50000 or 1521.
   f. Save your updates.
19. Optional: For DB2 systems, check the results by running db2cc from the DB2 server, or by performing the following steps from the database manager:
   a. On DB2, connect to `EVENT user db2inst1`, using **connect to CEI EVENT database** . The DB2 list table displays.
   b. Verify that the display shows a list of tables whose names start with CEI.

## Configuring security for CEI

If security is enabled for applications that are invoking the CEI emitter, then security must be configured for the service.

### Before you begin

- Stop the deployment manager and nodes for the Access Gateway cluster

### About this task

Complete the following steps on the deployment manager where the Access Gateway is deployed and synchronize the changes to the other nodes in the cluster.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Either turn off security for the bus used to send CEI messages or ensure that the user name and password used for authentication is the WebSphere Application Server Administrator.
3. Turn off security for the CEI bus.
   a. Under Service Integration select **Buses**.
   b. Under Buses click **CommonEventInfrastructure Bus**.
   c. Under Security, clear the **Secure** box.
   d. Click **Apply**.
   e. Click **Save** to save changes to the master configuration.
   f. Click **Save**.

4. To run with security for the CEI bus, ensure the authentication user name password is correct. The user name and password should be the WebSphere Application Server Administrator.
   a. Under Service Integration select **Buses**.
   b. Under Buses click **CommonEventInfrastructure Bus**.
   c. Under Related Items click **J2EE Connector Architecture (J2C) authentication data entries**.
   d. Under J2C Authentication data entries click the entry for ***node_name*/CommonEventInfrastructureJMSAuthAlias**, where *node_name* is the name of the node.
   e. Set the user name and password to the WebSphere Application Server Administrator user name and password.
   f. Click **Apply**.
   g. Click **Save** to save changes to the master configuration.
   h. Click **Save**.
5. There are several buses defined that should be checked for security and authentication values in the WebSphere ESB server.
   a. Under Service Integration select **Buses**.
   b. Under Buses, click each bus listed. Then either clear the **Secure** check box or follow the **J2EE Connector Architecture (J2C) Authentication data entries** link and verify the correct user name password for administrator is used.
6. Synchronize the changes to the other nodes in the cluster.

## Enabling the Fault and Alarm component Web service to use CEI

By default, each Service Platform component is configured not to use the CEI service. To use the CEI service, it must be enabled both for the Service Platform component and for WebSphere Application Server.

### Before you begin

- The TWSS Administration Console must be installed.
- The CEI service must be enabled for WebSphere Application Server.

### About this task

To use the CEI service, you must enable it for each Service Platform component.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.

3. In the Navigation pane, click **Web Services Platform**.
4. Click the cluster that you need to modify.
5. Click **faultalarm.ear**.
6. Click **Fault and Alarm Web Service**
7. For the **CEI Notifications Enabled** field, select `true`.
8. Click **OK**.

## Configuring a single network resource for service implementations

When a single network resource is used for admission control, you can enable it in the common setting for the service you decide to activate. In the TWSS Administration Console, the Network Resource plane field defines the network resource name.

### About this task

This configuration applies to any of the following Web service implementations:

- WAP Push over SMPP
- Parlay X SMS over SMPP
- Parlay X Multimedia Messaging over MM7
- Parlay X Terminal Location over MLP
- Parlay X Terminal Status over Parlay
- Parlay X Terminal Location over Parlay
- Parlay X SMS over Parlay
- Parlay X Call Handling over Parlay
- Parlay X Call Notification over Parlay
- Parlay X Third Party Call over Parlay
- Terminal Status and Presence over SIP/IMS
- Parlay X Call Notification over Parlay
- Parlay X Third Party Call over Parlay

In the following procedure, configuration steps are provided for the Parlay X SMS over SMPP Web service. To configure other Web services, substitute the path name found in the table.

*Table 34. Default network resource names and path names for the Web service implementations*

| Web service implementation | Default network resource name | Path name |
|---|---|---|
| WAP Push over SMPP | WAP10_PUSH_SMPP | **Web Services → WAPPush Service** |
| SMS over SMPP | PX21_SMS_SERVICE | **Web Services → SMS Service** |
| MMS over MM7 | PX21_MM_MM7 | **Web Services → IMS Multimedia Messaging Web Service** |
| Terminal Location over MLP | PX21_TL_MLP | **Web Services → Terminal Location Web Service** |
| Terminal Status over Parlay | PX21_TS_Parlay | **Web Services → Terminal Status Web Service** |

*Table 34. Default network resource names and path names for the Web service implementations (continued)*

| Web service implementation | Default network resource name | Path name |
|---|---|---|
| Terminal Location over Parlay | PX21_TL_Parlay | **Web Services** → **Terminal Location Web Service** |
| SMS over Parlay | PX21_SMS_SERVICE | **Web Services** → **SMS Service** |
| Call Handling over Parlay | PX21_CH_Parlay | **Web Services** → **Call Handling Web Service** |
| Call Notification over Parlay | PX21_CN_Parlay | **Web Services** → **Call Notification Web Service** |

1. In the TWSS Administration Console, navigate to the SMS Service Settings page.
2. Populate the following fields on the Runtime tab of the **General Properties** section:
   - **Purge Enabled**: **False** is the default option in the pull-down.
   - **Purge Interval**: The number **10** is the default value.
   - **Purge Block**: The number **1000** is the default value.
   - **Service Enabled**: **True** is the default option in the pull-down.
   - **Network Resource Name**: The user enters the **Network Resource Name** used for traffic shaping.
3. Click **OK** to enable the network resource name.

## Configuring integration with other parts of the network

Other configuration options exist for using the Telecom Web Services Server components with certain other elements within your telecommunications network.

### Provisioning the Inter Operator Identifier (IOI)

The charge header support vector utility provides a utility class for handling the SIP (session initiation protocol) messaging for charging interactions. Before you can use the utility, you must provision the Inter Operator Identifier (IOI). This is done using the TWSS Administration Console.

#### About this task

To provision the IOI, complete the following steps.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)

   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services**.
4. Click ***Scope_name*** in the list.
5. Click the name of the EAR file: **IMS Third Party Call.ear**.
6. In the component configuration, click the Third Party Call Web service.
7. Set the value for **Inter Operator Identifier** to the domain name for the S-CSCF, for example `example.com`.
8. Click **OK**.

## Finding the IBM XDMS XCAP root

If you do not know what the IBM XDMS XCAP root is, you can find it in the TWSS Administration Console. The default value for XCAP root is `http://<hostname>:port/services`.

### Before you begin

You must have completed the following steps:
- Deployed IBM XDMS
- Started the application server

1. Log into the TWSS Administration Console for the server where IBM XDMS is deployed.
2. Click **Security** → **Global security** and enable both administrative security and application security.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Security** → **Secure administration, applications, and infrastructure**.
3. In the navigation pane click **Resources** → **Resource Environment** → **Resource Environment Providers**.
4. On the Resource Environment Providers page, select the **ibm-xdms** Resource Environment Provider that is in the same scope as `IBMSharedListXDMS.ear`. For example, **Cluster = cluster 1**
5. On the Configuration page for ibm-xdms, select **Additional Properties Custom Properties**.
6. The XCAP root will be listed in the **xcapRoot property** list.

# Chapter 5. Administering WebSphere Telecom Web Services Server

You can use different user interfaces to manage and maintain the Telecom Web Services Server.

- Use the TWSS Administration Console plug-in, part of the Integrated Solutions Console, to configure the Web service implementations and the Service Platform components and to administer MBeans that are included with the Web service implementations.
- Use the Service Policy Manager console to manage policy data for the Web service implementations.
- Use the Integrated Solutions Console and the Parlay Administration Console plug-ins to configure integration with the Parlay gateway.

## Stopping and starting the server

After making changes to the server configuration, you must restart the application server.

### About this task

In a clustered environment, some tasks require you to restart the deployment manager for changes to take effect. To stop the deployment manager, you must stop all application servers, all node agents, and then the deployment manager. To restart the deployment manager, you must start the deployment manager, all node agents, and then the cluster (which starts all application servers).

The following instructions describe how to stop and restart resources both from the Integrated Solutions Console and from a command-line prompt.

**Note:** `was_profile_root` is the directory for a WebSphere Application Server Network Deployment profile called *profile_name*. By default, this directory is:

AIX `/usr/IBM/WebSphere/AppServer/profiles/`*`profile_name`*

Linux `/opt/IBM/WebSphere/AppServer/profiles/`*`profile_name`*

`/opt/IBM/WebSphere/AppServer/profiles/`*`profile_name`*

### Stopping a cluster

#### About this task

When you stop a cluster, all application servers on the cluster are stopped.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

**Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Stop the cluster:
   a. In the Integrated Solutions Console, click **Servers** → **Clusters** → **WebSphere application server clusters**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Clusters**.
   b. Select the check box associated with the name of the cluster.
   c. Click **Stop**.

# Stopping a server (console)

## About this task

Stopping an application server stops all applications automatically.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Stop the application server:
   a. In the Integrated Solutions Console, click **Servers** → **Server Types** → **WebSphere application servers**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Application servers**.
   b. Select the check box associated with the name of the server.
   c. Click **Stop**.

# Stopping a server (command line)

Run the following command:

AIX
```
was_profile_root/bin/stopServer.sh server_name -username user_name
-password password
```

Linux
```
was_profile_root/bin/stopServer.sh server_name -username user_name
-password password
```

```
was_profile_root/bin/stopServer.sh server_name -username user_name
-password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of the application server profile (for example, AppSrv01).

*server_name* is name of the application server.

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

## Stopping the node agent (console)

### About this task

When stopping the deployment manager and application servers, you must also stop the node agents. If you are stopping a cluster, you must stop all node agents.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Stop one or more nodes:
   a. In the Integrated Solutions Console, click **System administration** → **Node agents**.
   b. Select the check boxes associated with each node.
   c. Click **Stop**.

## Stopping the node agent (command line)

Run the following command:

AIX
```
was_profile_root/bin/stopNode.sh -username user_name -password password
```

Linux
```
was_profile_root/bin/stopNode.sh -username user_name -password password
```

```
was_profile_root/bin/stopNode.sh -username user_name -password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of a federated node profile (for example, Custom01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name.*

## Stopping the deployment manager (console)

### About this task

When stopping the servers and node agents in a cluster, you must also stop the deployment manager. When the deployment manager is stopped, you will not be able to access the Integrated Solutions Console.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Stop the deployment manager:
   a. In the Integrated Solutions Console, click **System administration** → **Deployment manager**.
   b. Click **Stop**.

## Stopping the deployment manager (command line)

Run the following command:

AIX

```
was_profile_root/bin/stopManager.sh -username user_name -password password
```

Linux

```
was_profile_root/bin/stopManager.sh -username user_name -password password
```

```
was_profile_root/bin/stopManager.sh -username user_name -password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *`was_profile_root`* path contains the name of the deployment manager profile (for example, Dmgr01).

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

## Starting the deployment manager

### About this task

Start the deployment manager before starting the node agents and application servers. When the deployment manager is started, you will have access to the Integrated Solutions Console.

Run the following command:

AIX *was_profile_root*/bin/startManager.sh

Linux *was_profile_root*/bin/startManager.sh

*was_profile_root*/bin/startManager.sh

Where:

The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).

## Starting the node agents

### Before you begin

After starting the deployment manager, you must start the node agents before you can start the cluster or the application server.

Run the following command:

AIX *was_profile_root*/bin/startNode.sh

Linux *was_profile_root*/bin/startNode.sh

*was_profile_root*/bin/startNode.sh

Where:

The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).

## Starting a cluster

### About this task

When you start a cluster, all application servers on the cluster are started.

1. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
    b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    c. Click **Log in**.
2. Start the cluster:
    a. In the Integrated Solutions Console, click **Servers** → **Clusters** → **WebSphere application server clusters**.

       **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Clusters**.
    b. Select the check box associated with the name of the cluster.
    c. Click **Start**.

## Starting a server (console)

### About this task

Applications that were running when the server was stopped are restarted automatically.

1. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

       **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
    b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
    c. Click **Log in**.
2. Start the application server:
    a. In the Integrated Solutions Console, click **Servers** → **Server Types** → **WebSphere application servers**.

       **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Application servers**.
    b. Select the check box associated with the name of the server.
    c. Click **Start**.

## Starting a server (command line)

Run the following command:

AIX

```
was_profile_root/bin/startServer.sh server_name -username user_name
-password password
```

Linux

```
was_profile_root/bin/startServer.sh server_name -username user_name
-password password
```

```
was_profile_root/bin/startServer.sh server_name -username user_name
-password password
```

**Note:** The `user_name` and `password` parameters are required only when security is enabled.

Where:

The *was_profile_root* path contains the name of the application server profile (for example, AppSrv01).

*server_name* is name of the application server.

*user_name* represents your WebSphere Application Server administrator user ID.

*password* represents the password associated with your *user_name*.

## Restarting applications

You can restart one or more applications at a time using the Integrated Solutions Console.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Applications** → **Enterprise Applications**.
3. Select the check box adjacent to each application that you want to stop. You can select more than one application.
4. Click **Stop**. The Application Status column shows that the application is stopped.
5. Select the check box adjacent to each application that you want to start. You can select more than one application.
6. Click **Start**. The Application Status column shows that the application is started.

## Modifying logging

Use the Integrated Solutions Console to specify how data is logged, where the log data is stored, and the output format to use for log data.

### About this task

You can modify the general properties of each log, which specifies the output type or location of the log. Use the following steps to adjust the properties for each log type:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)

   c. Click **Log in**.
2. In the navigation pane, click **Servers** → **Server Types** → **WebSphere application servers**.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Application servers**.
3. Click the name of the server you want to manage.
4. Under Troubleshooting, click **Logging and Tracing**.
5. Click one of the log types. Then click the Configuration tab to make a static change to the system log configuration, or click the Runtime tab to change the configuration dynamically.

   **Note:** Separate logs for each log type exist for all Java virtual machines (JVMs) on a node, including all application servers and their node agent, if present, as well as for a deployment manager in its own logs directory.

   Here is a list of the available log types:

| Option | Description |
|---|---|
| **Diagnostic Trace** | Provides information in the `trace.log` about how the WebSphere Application Server components run. |
| **JVM Logs** | Used to view and modify the settings for the Java Virtual Machine (JVM). The `System.out` log (`SystemOut.log`) is used to monitor the health of the WebSphere Application Server. The `System.err` log (`SystemErr.log`) contains exception stack trace information used to perform problem analysis. |
| **Process Logs** | Created when redirecting the standard out and standard error streams of a process to independent log files, the `native_stdout.log` and `native_stderr.log`, respectively. |
| **IBM Service Logs** | Also known as the activity log. Records the WebSphere Application Server messages that are written to the `System.out` stream and special messages that contain extended service information that you can use to analyze problems. |
| **Change Log Detail Levels** | Controls which events are processed by Java logging, by using log levels. You can assign logging levels to individual trace loggers or to trace groups. (Trace loggers and groups are listed in the topic *Trace loggers*.) |

6. When you are finished making your changes, click **Apply**.
7. Click **OK**.
8. Click **Save** to save changes to the master configuration.
9. Optional: If you made a static change to the configuration, restart the application for your changes to take effect.

# Monitoring system performance using WebSphere PMI

WebSphere Performance Monitoring Infrastructure (PMI) provides data that you can use to monitor and tune your application server's performance. PMI consists of several metrics (indicators and counters), each of which provides statistics about a specific aspect of the system's performance. You can enable and disable the metrics according to your specific needs.

The metrics used by WebSphere PMI are sometimes referred to as *key performance indicators*.

For more information about using WebSphere PMI, see the topic *Performance Monitoring Infrastructure (PMI)* in the WebSphere Application Server 7.0 Information Center.

## Enabling performance monitoring

Use the Integrated Solutions Console to enable performance monitoring for your application server.

### About this task

Complete the following steps to enable PMI-based performance monitoring.

For a full list of supported metrics, refer to the topic *Performance metrics*.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Monitoring and Tuning** → **Performance Monitoring Infrastructure (PMI)**.
3. Select the application server instance.
4. In the Configuration tab, under General Properties, select the check box labeled **Enable Performance Monitoring Infrastructure (PMI)**.
5. Under Currently monitored statistic set, select **Custom**. Then click the **Custom** link and use the dialog to specify the list of metrics you want to enable.
6. Click **OK**.
7. Click **Save** to save changes to the master configuration.
8. In a clustered configuration, repeat steps 3 through 7 for each application server.
9. Restart the server.

## Disabling performance monitoring

For best system performance, it is a good practice to disable PMI metrics when the data is no longer needed.

### About this task

Complete the following steps to disable PMI-based performance metrics that you no longer need.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Monitoring and Tuning** → **Performance Monitoring Infrastructure (PMI)**.
3. Select the application server instance. For a clustered configuration, select all of the application server instances.
4. In the Configuration tab, under General Properties, deselect the check box labeled **Enable Performance Monitoring Infrastructure (PMI)**.
5. Click **OK**.
6. Click **Save** to save changes to the master configuration.
7. Restart the server.

## Performance metrics

WebSphere Performance Monitoring Infrastructure (PMI) indicators provide a comprehensive set of data to help explain the behavior of applications and the resources they consume.

The following tables list the metrics (indicators and counters) that are supported for the Telecom Web Services Server product.

*Table 35. Performance metrics*

| Name | Description |
|---|---|
| TransactionIdentifierMediatorExecutionTime | Execution time, in milliseconds, of the Transaction Identifier mediation primitive |
| SPMResponseTime | Response time, in milliseconds, for the Policy Retrieval mediation primitive call to the Service Policy Manager |
| GLSResponseTime | Response time, in milliseconds, for the Group Resolution mediation primitive call to the group list server |

*Table 35. Performance metrics (continued)*

| Name | Description |
|---|---|
| TransactionRecordTime | Database access time, in milliseconds, for the Transaction Recorder mediation primitive |
| NetStatsRecordTime | Database access time, in milliseconds, for the Network Statistics mediation primitive |
| FlowLatencyTime | Total time, in milliseconds, including global handler and all mediator processing, up to the call to the backend Web service implementation |
| ServiceImplResponseTime | Response time, in milliseconds, from the backend call to the Web Service implementation |
| FlowResponseTime | Total time, in milliseconds, to execute the flow and return from the flow (from inside the Access Gateway) |
| ExportAttachmentExecutionTime | Execution time, in milliseconds, for the export attachment handler |
| ImportAttachmentExecutionTime | Execution time, in milliseconds, for the import attachment handler |
| TransactionMediatorExecutionTime | Execution time, in milliseconds, for the Transaction Recorder mediation primitive |
| MsgElementRemoverExecutionTime | Execution time, in milliseconds, for the Message Element Remover mediation primitive |
| PolicyMediatorExecutionTime | Execution time, in milliseconds, for the Policy mediation primitive |
| GroupResolutionMediatorExecutionTime | Execution time, in milliseconds, for the Group Resolution mediation primitive |
| NetStatsMediatorExecutionTime | Execution time, in milliseconds, for the Network Statistics mediation primitive |
| JMXNotificationExecutionTime | Execution time, in milliseconds, for the JMX Notification mediation primitive |
| ServiceAuthMediatorExecutionTime | Execution time, in milliseconds, for the Service Authorization mediation primitive |
| ServiceInvocationMediatorExecutionTime | Execution time, in milliseconds, for the Service Invocation mediation primitive |
| SLAClusterMediatorExecutionTime | Execution time, in milliseconds, for the SLA Enforcement mediation primitive |
| IncomingFlowTransactionsPerSecond | Number of incoming transactions per second in the Access Gateway flow |
| OutgoingFlowTransactionsPerSecond | Number of outgoing transactions per second in the Access Gateway flow |
| FlowExecutionTime | Total execution time, in milliseconds, of the Access Gateway flow |
| ServiceStartupTime | Time the service flow is initialized, which occurs on the first request |
| RequestsReceived | Number of flow requests received |
| RequestsSuccessful | Number of flow requests processed successfully |

*Table 35. Performance metrics (continued)*

| Name | Description |
|---|---|
| RequestsFailed | Number of flow requests that failed |
| RequestsAuthorizationAccepted | Number of flow requests authorized |
| RequestsAuthorizationDenied | Number of flow requests for which authorization is denied |

## Administering the Service Policy Manager

The Service Policy Manager (SPM) manages hierarchical policy definitions and service subscriptions for requesters. The key benefits of this component are a common database for the management of requesters, services and subscriptions; support for attaching policies to managed entities; and hierarchical resolution of policies enabling rich policy resolution with easy administration.

A service policy defines a piece of runtime configuration data for a particular service. Service policies can be associated directly with requesters (as part of a subscription) to provide personalization of service delivery. It is defined as a name and a value, with the interpretation of the service policy performed by message processing or application logic.

### Accessing SPM

Your applications access the Service Policy Manager (SPM) through a Web service, which enables the SPM to be deployed and managed independent of the applications that utilize it.

Because the SPM uses a Web services interface, it must be integrated with your network environment. The SPM recognizes policies as data, which it manages and stores, but the services that define and use these data are solely responsible for interpreting it.

As an alternative to invoking Web service operations, the SPM console provides a graphical interface with which you can manage entities like requesters, services, subscriptions, and policies. For details on its use, refer to the topic *Launching the Service Policy Manager console*.

There is also a scripting (MBean) interface for batch operations, such as adding policies for Web services.

### Default policies

Policy information in the incoming request is overridden by a returned policy configuration for the Service Policy Manager (SPM). This applies to any matching policies. The non-matching policies will not be removed from the incoming request.

### Subscriptions, services, and subscribers

A subscription is an association of a requester to a service or a specific service operation. Within a subscription, a set of policies may be defined that characterize the behavior of the use of a service by the requester. A requester which has a subscription is a subscriber. The Service Policy Manager (SPM) provides a storage capability and access mechanism to enable the definition of requesters, services and subscriptions that associate services with requesters. Policies may be attached to each of these, and are resolved using a hierarchical algorithm.

For example, service level agreements, which represent the terms of a contract between the entity owning the requester and the provider, may be expressed as a set of policies that manage the use of the service.

**Note:** This component is not a subscriber data or service data management system; it is a subscription policy management system. Management of subscriber data and service data is outside the scope of this component.

## Launching the Service Policy Manager console

The Service Policy Manager (SPM) console provides a graphical interface for creating, viewing, modifying, and deleting entities like requesters, services, subscriptions, and policies.

### Before you begin

Before you can use the SPM console, it must be installed on the deployment manager.

Because the SPM console communicates with the SPM runtime component, the SPM runtime component must be up and running.

### About this task

**Note:** If you are using the SPM console with the Firefox browser, Firefox version 3.0 is strongly recommended.

To use the SPM console, follow these steps:

1. Log in to the console.
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/spm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password.
   c. Click **Log in**.
2. In the Runtime Connection window, specify the instance of the Service Policy Manager runtime component to which you want to connect.
   a. From the list, select the URL for the SPM runtime instance. If the runtime instance is not listed, select the text box and type a URL. The list displays URLs for the three most recent connections.
   b. Enter a user name and password for establishing the connection. These credentials are used for connecting to the SPM runtime instance. They are not necessarily the same credentials you used to log in to the console.

c. Optional: Select a **Connection Timeout** value. This value determines how long the SPM console will wait for a response from the runtime instance before terminating the connection.
d. Click **Connect**.

3. In the main console window, click the portlet you want to manage, for example **Requester Administration** or **Policy Administration**. Information about managing the portlets is found in the portion of this information center titled *Using the features of the Service Policy Manager*.

## Using the features of the Service Policy Manager

Accessed either through a Web service interface or in the graphical console, the Service Policy Manager provides a tool for managing requesters, services, subscriptions, policies, and other entities.

### Web service integration

Using a Web service interface called `ServicePolicyAccess`, you can expose the Service Policy Manager (or a custom-developed alternative) as a Web service that can be used by applications to retrieve service policies.

#### Using ServicePolicyAccess

Telecom Web Services Server provides the Service Policy Manager component, which uses a Web service interface to access the policies. (The Access Gateway component references the `ServicePolicyAccess` interface to call the `getServicePolicies` operation.)

You have the option of replacing the Service Policy Manager with your own custom-developed alternative. Start with the WSDL for `ServicePolicyAccess` and create your own implementation of the Web service interface. For more information, refer to the topic *WSDL documentation for WebSphere Telecom Web Services Server*.

#### General integration capabilities

The Service Policy Manager component provides both a Web service interface and a user interface that allows scripting through JMX and wsadmin, using such languages as Jython or Python.

The Web service interface is divided into two different application programming interfaces (APIs):

- Access interface
- Administrative interfaces

These capabilities are exposed through remotely accessible interfaces. The runtime exposes two sets of interfaces: an access interface that is used to retrieve policy information for a given requester, service, and operation, along with a set of administrative interfaces that allows for the management of requesters, services, subscriptions, data types, and policy values.

The Service Policy Manager resolves policy values using a hierarchical algorithm; requesters and services can be organized into a tree hierarchy that allows for groupings of requesters and services. Both subscriptions and policies can then be set within the tree scope. The lookup process is hierarchical, allowing requesters and services that are lower in the tree to inherit subscriptions and policy values from their parents. The console builds on top of the runtime to provide graphical portlet capabilities for managing service policy manager entities. The console can be deployed either standalone or within a full portal runtime.

The Web service admin scripting is the preferred bulk loading or initializing process of the Service Policy Manager. This allows for administering Service Policy Manager definitions from the command-line interface using scripting, or a JMX browser for discovery. Additionally, JMX can be used for discovery purposes to determine whether the Service Policy Manager has been deployed in the same WebSphere Application Server installation.

Using wsadmin scripting in Jython mode is recommended. The following topics in the WebSphere Application Server information center provide a good overview for how to use wsadmin scripting. Both are located within the major topic *Scripting the application serving environment (wsadmin)*.

- *Getting started with scripting*, which has subtopics for using the wsadmin scripting objects and starting the wsadmin scripting client
- *Scripting and command line reference material*

## Example: Accessing the RequesterAdministration MBean

The following is an example of Jython code that accesses the **RequesterAdministration** MBean to perform a query:

```
$ ./wsadmin.sh –lang jython
mbean_list = AdminControl.queryMBeans("type=RequesterAdministration,*") length = mbean_list.size()
if length > 0:
inst = mbean_list[0]
# Just use the first one returned
obj_name = str(inst.getObjectName())
# Print out all attributes
print AdminControl.getAttributes(obj_name, None)
```

## Example: Accessing the PolicyAdministration MBean

The following is an example of Jython code that accesses the **PolicyAdministration** MBean to update a policy:

```
$ ./wsadmin.sh -lang jython -f polinit.py

import sys.argv
import getopt
from com.ibm.twss.spm.admin.common import ScopedPolicy
from com.ibm.twss.spm.admin.common import PolicyValue
from java.util import Properties

from pytwss.mbean import spm_init, mbean_utils
from pytwss.utils import _, log

A_SERVICE = 'http://www.csapi.org/wsdl/parlayx/A/v2_3/interface'
SVC_GROUP = 'AService'
SERVICE_IMPL = 'PX21_A_PARLAY'

def configurePolicies(response_properties):
 req_scope = 'ALL'
 svc_scope = SVC_GROUP
 op_scope = 'ALL'

 svc_admin = spm_init.ServiceAdministration()
 policy_admin = spm_init.PolicyAdministration()

 spm_init.registerServiceImplementation(
  svc_admin, A_SERVICE, SERVICE_IMPL, 1,
  'AService description')

        sp_root_url = response_properties.get("ServicePlatformRootURL")

 policies = [
  ScopedPolicy(
   requesterIdentifier=req_scope,
   serviceIdentifier=svc_scope,
   operation=op_scope,
   name='service.config.StatusRetainTime',
   value=PolicyValue(
    value='2000',
    type='Numeric'
   )
  ),
  ScopedPolicy(
   requesterIdentifier=req_scope,
   serviceIdentifier=SERVICE_IMPL,
   operation=op_scope,
   name='service.Endpoint',
   value=PolicyValue(
    value=sp_root_url + "/TWSS/ParlayX21/AService/Parlay/services/AService",
    type='String'
   ),
  ),
 ]
 spm_init.createPolicies(policy_admin, policies)

        return "SUCCESS";

if __name__ == 'main':
   if len(sys.argv) > 0:
      sp_path = sys.argv[0]
   else:
      sp_path = "http://localhost:9080/"
   p = Properties()
   p.put("ServicePlatformRootURL", sp_path)
   configurePolicies(p)
```

## Role based authorization

When the Service Policy Manager component functions as a Web service, it can be used in two different ways–depending on the WebSphere Application Server role that is assigned to the user.

The roles are as follows:

- **PolicyAccessor**: The user can retrieve policy information from the access interface, but the user cannot access any administrative functions.
- **PolicyAdministrator** : The user can use the access interface and can access all administrative functions.

## Policy administration Web services

You can administer the data stored by the Service Policy Manager through a Web service interface.

Documentation for the Web service interface is included in the topic *WSDL documentation*, in the Reference section of this information center.

## Service policy naming conventions

Service policies use a hierarchical naming scheme to describe the policy name. The notion of hierarchy in the name is only to provide additional meaning to the policy names to make them easier to read. The actual hierarchical capabilities are provided by the Service Policy Manager runtime component.

The naming convention does not affect a policy's position in the Service Policy Manager inheritance hierarchy. Each name can have multiple parts, where each part is separated by a period (**.**) delimiter.

The following categories are defined:

- Category: **message** (message processing service policies)

  **message.capacity**
  Policies for capacity measurements

  **message.intercept**
  Policies for message intercept

  **message.sla**
  Policies for SLA measurements

  **message.statistics**
  Policies for network statistics

  **message.privacy**
  Policies for privacy
- Category: **requester** (requester policies)

  **requester.operation**
  Policies to apply to requester use of operations

  **requester.trace**
  Policies related to tracing requests by requester
- Category: **service** (service policies)

  **service.config**
  Policies related to service configuration

  **service.custom**
  Policies defined for Web service implementations

  **service.standard**
  Policies defined by Web services specifications
- Category: **customer** (values that are customer unique and are not recorded in the service usage record, but are made available to plug-in points)
- Category: **usage** (values that are to be passed through to the service usage record)

A Service Policy Manager installation includes a default requester hierarchy and a default service hierarchy. They are enumerated in the following sections.

## Requester hierarchy

For the requester hierarchy, the following interface definitions are initialized by default.

A group named ALL, containing all of these definitions, is also initialized.

Requester: unauthenticated

Group: authenticated

- Requester: anonymous.invalid
- Requester: notification

## Service hierarchy

For the service hierarchy, the following interface definitions are initialized by default. Use these definitions to help you register the Web service implementations.

A group named ALL, containing all of these definitions, is also initialized.

Group: WAP Push
- Service: `http://www.ibm.com/wsdl/wappush/send/v1_0/interface`

Group: Parlay X
- Group: Third Party Call
  - Service: `http://www.csapi.org/wsdl/parlayx/third_party_call/v2_3/interface`
- Group: Call Notification
  - Service: `http://www.csapi.org/wsdl/parlayx/call_notification/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/call_notification/notification_manager/v2_3/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/call_direction/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/call_direction/notification_manager/v2_3/interface`
- Group: SMS
  - Service: `http://www.csapi.org/wsdl/parlayx/sms/notification_manager/v2_3/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/sms/send/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/sms/receive/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/sms/notification/v2_2/interface`
- Group: MMS
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/notification_manager/v2_5/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/send/v2_4/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/receive/v2_4/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/notification/v2_4/interface`
- Group: Payment
  - Service: `http://www.csapi.org/wsdl/parlayx/payment/amount_charging/v2_1/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/payment/reserve_amount_charging/v2_1/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/payment/reserve_volume_charging/v2_2/interface`
- Group: Account Management
  - Service: `http://www.csapi.org/wsdl/parlayx/account_management/v2_2/interface`
- Group: Terminal Status

  - Service: `http://www.csapi.org/wsdl/parlayx/terminal_status/notification_manager/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/terminal_status/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/terminal_status/notification/v2_2/interface`
- Group: Terminal Location
  - Service: `http://www.csapi.org/wsdl/parlayx/terminal_location/notification_manager/v2_3/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/terminal_location/v2_2/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/terminal_location/notification/v2_2/interface`
- Group: Call Handling
  - Service: `http://www.csapi.org/wsdl/parlayx/call_handling/v2_3/interface`
- Group: Audio Call
  - Service: `http://www.csapi.org/wsdl/parlayx/audio_call/v2_1/interface`
- Group: Multimedia Conference
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/notification_manager/v2_5/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/send/v2_4/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/receive/v2_4/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/multimedia_messaging/notification/v2_4/interface`
- Group: Address List Management
  - Service: `http://www.csapi.org/wsdl/parlayx/group_mgmt/v2_1/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/group/v2_1/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/group_member/v2_1/interface`
- Group: Presence
  - Service: `http://www.csapi.org/wsdl/parlayx/presence/supplier/v2_3/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/presence/consumer/v2_3/interface`
  - Service: `http://www.csapi.org/wsdl/parlayx/presence/notification/v2_3/interface`

**Note:** Registration of a Web service implementation is performed when installing a Web service implementation through the First Steps script auto configuration facility.

## Service policy types and elements

The Service Policy Manager enables you to mange policies globally and individually. It stores predefined values to optimize performance, and you can use it to create default operation policies.

Service policies are managed as a logical hierarchy, allowing values to be specified at different levels and to have the values coalesced in a predictable manner to determine the values that will be used for a particular message. This provides wide flexibility in the definition of default values, and the ability to adapt to a wide variety of business and technical requirements for the definition of access rights, service level agreements, and service adaptations.

| Name | Description |
|---|---|
| Service policies | Service policies typically have a multipart name, where each part is separated by a period (**.**) delimiter. |
| Global service policies | Global service policies can be defined by you, and they have scope over all deployed services. They may be created, updated, queried and deleted by you. Global service policies are presented in all other subscription management panels, allowing you to override them. Global service policies are created by specifying the special service value *ALL* which is synonymous with all services. |
| Predefined values | For the fast ordered retrieval of hierarchical policies, the Service Policy Manager stores the predefined values in the database as a number string.<br>For example, the requesters `ALL` and `unauthenticated` are stored as strings `0` and `00`, respectively. The service `ALL` and operation `ALL` are stored as string `0`. |

## Predefined values

The Service Policy Manager provides several predefined policy types and two predefined value special requesters: `ALL` and `unauthenticated`.

### Predefined policy types

A policy type associates validation logic with a policy attribute. The validation logic decides the contents which are allowed as the associated attribute value.

In the Service Policy Manager, policy types serve to validate only the form of the value contents, not the semantics. Semantics are intended to be interpreted by the services that use those policy attributes. In implementation, the form factor will be enforced through the regular expressions that validate the content of the policy value string. This can be used to enforce basic types such as integers and floats, as well as more complex strings such as URIs.

The predefined policy types are as follows:

- `ReadOnlyString`: not editable, read-only string, not empty
- `RequiredString`: string, not empty
- `String`: string, may be empty
- `Numeric`: positive numeric value
- `RequiredNumeric`: positive numeric value, not empty
- `Boolean`: true or false
- `URI`: URI; for example, http://www.ibm.com
- `Password`: a string value containing a password, which is encrypted and obscured.

- `TimeSpecification`: value for the TimeMetric data structure
- `ReportingInterval`: reporting interval for notifications
- `IntegerList`: positive numeric comma-separated list, not empty
- `StringList`: comma-separated list, not empty

### Operations

The Service Policy Manager provides one predefined special operation `ALL` which is synonymous with all operations.

You can specify the operation as `ALL` to create default operation policies, which Service Policy Manager will apply to all existing and future operations within the service.

## Enrolling a new subscriber

Each requester (third-party application) must be authorized to access a set of services, and this is done by means of a subscription. Use the Service Policy Manager console to define requesters and set up subscriptions.

### About this task

A subscription must be created whether you are unauthenticated or not. Subscriptions can be inherited if they exists higher in the hierarchy. If a subscription is created for the requester *ALL*, then all requesters will inherit that subscription. Inherited subscriptions can be overridden if a particular requester or group wishes to subscribe to a different implementation for a given service.

To set up a new subscription, complete the following steps:

1. Launch the Service Policy Manager (SPM) console.

   **Note:** If you are using the SPM console with the Firefox browser, Firefox version 3.0 is strongly recommended.
2. Using the **Requester** view, create a requester. Each requester is enabled and has a descriptive string. This requester identity must also be known to the WebSphere user registry and be authorized through the Integrated Solutions Console to have the PolicyAccessor role.
3. Using the **Subscriptions** view, create a subscription between the requester and the necessary services and operations. Set up the subscription for all services specified as `GROUP`, and for all operations specified as `ALL`.

   **Note:** The Service Policy Manager access Web service `GetServicePolicies` operation returns policies associated with a subscription. But if the subscription does not exist, a `ServiceException` is generated.

   To set up the user name and password, you can use the following policies: `service.notification.username` and `service.notification.password`. The policies should be created for the appropriate requester, for the appropriate service, and for all of the operations. (Operations will be specified as `ALL`

   **Note:** Password policies should be of the type `password`, and not `string`).

# Administering Service Platform components

Use the TWSS Administration Console to administer the Service Platform components and adjust configuration settings.

**Note:** The recommended method of deploying the Service Platform components is to use local Web service invocations and WebSphere Application Server in-process optimization.

## Administering the Admission Control component Web service

The Admission Control component Web service protects the platform on which the Service Platform components run from overload conditions and can control the types of traffic admitted into the platform. The Admission Control component Web service offers two types of admission limit enforcement: locally tracked limits and cluster-wide tracked limits.

Local limits are useful for bounding the rate of requests accepted on an individual server and are tracked only using local request information. Cluster limits are useful for bounding the rate of requests accepted across the cluster as a whole. A distributed reservation algorithm is used to allocate rate to individual cluster members as needed, while ensuring that the total limit across the cluster is exceeded. This approach can handle different distributions of load across the cluster, which may arise as a result of the use of session affinity (directed all requests that are part of a session to the same application server) for converged HTTP and SIP applications.

The Admission Control component Web service allows for setting hierarchical rate limits for service and operation traffic within the TWSS Administration Console. Services and operations form a hierarchy, where each service has a set of child operations. Limits set at the server level constrain the rate of requests for all operations executed against that service, regardless of the distribution of requests to the individual operations. Limits set at the operation level limit the rate of requests for the particular operation. Request rates are expressed in tokens per second. Weights can be set for individual services and operations within the TWSS Administration Console. When determining whether to admit an operation for a particular service, a total token weighting for the given operation is calculated using the formula: (service weight * operation weight). This total weighting represents the number of tokens that will be consumed by admitting this request.

Hierarchical limits in the Admission Control component Web service are enforced using a sliding window, rate limiting bucket algorithm. The algorithm works as follows: the first request received with a positive weighting from a quiesce (idle with no active sliding window) state begins a sliding window over a one-second interval until the next quiesce state. Requests may arrive at any rate within the window as long as the limit is not exceeded. This restricts the average rate of Web service requests, regardless of the characteristics of the incoming flow of traffic. Requests with higher weighting will consume more tokens against the limit than requests with lower weighting. Requests with a weight of zero are automatically admitted, regardless of the limit.

### Reservation scheme

A reservation scheme is used to allocate rate amongst members of a cluster to enforce cluster-wide limits. A single coordinator is elected amongst the cluster members. The coordinator maintains reservation information for each cluster member against the cluster-wide limits. Cluster members send reservation requests as needed, such as when insufficient rate is currently allocated, to attempt to admit incoming requests. When allocated rate is no longer needed, a release request is sent to the coordinator relinquishing rate. To determine when to release rate, a rate estimator is associated with each operation. A running average of current traffic is maintained and used to determine when rate is no longer needed. Rate is released gradually (every second) in chunks that are a percentage of cluster-wide rate limit.

Each cluster member maintains a local hierarchy of rate limiting buckets that are used to calculate request admission. Reservations for additional rate capacity are made against each operation as more rate is needed. When additional rate has been reserved, the allocate rate is added to the operation rate limiting bucket and the parent service rate limiting bucket. As such, the rate limit for the service bucket will be the sum of the rates of its child operation buckets. A reservation request may be denied at any time due to lack of sufficient rate at the service or operation level. When a node member receives a rejected reservation request, it will enter a silence period. This silence period will suppress additional reservation requests, reducing inter-cluster traffic and allowing the network to settle before attempting additional reservation. In steady state, this minimizes the inter-cluster messaging required.

Upon failover of the node housing the cluster coordinator, a new coordinator is elected. A message is resent to all the nodes resetting their reservation status. This results in a ″reset″ of the algorithm without interrupting service. Inter-cluster traffic may increase temporarily while the algorithm rebuilds state to reduce inter-cluster messaging.

### Guidelines for choosing limits

There are no restrictions for choosing local and cluster limits. Local limits are assumed to be typically be smaller than cluster limits, although this is not a hard requirement.

**Note:** For cluster-wide limits, reservation requests are only made on an as-needed basis. Each request will ask for enough rate limit to accommodate the inbound request, which results in a reservation request that matches the token weight of the request. Clusters are typically fronted by a round-robin load-balancer. Thus, when running at capacity, it is possible that an individual node will get allocated a chunk of rate during a round robin spray and the other remaining members be denied. This may not match expectations across the cluster. For example, consider a cluster with three members, with a rate limit for an operation of 30 tokens per second across the cluster and each cluster member allocated 9 tokens per second. This leaves 3 tokens per second worth of rate to reserve. If an additional spray of requests comes on top of the 27 tokens per second rate across the cluster, then the first request in the spray will result in the first server getting allocated the 3 tokens per second and the remainder being denied additional capacity (and thus rejecting the request).

## Administering the Traffic Shaping component Web service

The Traffic Shaping component Web service controls the flow of outbound traffic from Service Platform components towards network resources. The Traffic Shaping component Web service implements a distributed token bucket algorithm that allows for specifying the maximum burst and maximum sustained average rate of traffic emitted from the cluster.

Burst traffic refers to Web service requests that are separated by short intervals as compared to the average rate of traffic and are commonly considered spikes in the traffic. The sustained average rate refers to the sustained traffic throughput allowed once all burst has been exhausted. The sustained average rate is commonly thought of as the steady state traffic rate. Traffic shaping applies across the cluster only (not locally) and is tracked on a per-resource basis. Multiple service implementations may refer to the same network resource name and thus share the same traffic limits and consumption tracking. A distributed reservation algorithm is used to allocate rate to individual cluster members as needed, while ensuring that the total limit across the cluster is not This approach can handle different distributions of load across the cluster, which may arise as a result of the use of session affinity (directing all requests that are part of a session to the same application server) for converged HTTP/SIP applications.

The Traffic Shaping component Web service should be called by service implementations prior to the generation of traffic. Each invocation to traffic shaping takes a token weighting representing the amount of traffic (or cost) of the action that the service implementation is about to initiate towards the network resource. The calculation of the token weighting is service implementation-specific and may correspond to the number of messages generated, number of operation targets, and so forth. This weighting corresponds to the number of tokens consumed from the distributed token bucket in admitting the request. This implementation does not perform any request queueing; the traffic shaping component merely indicates whether sufficient rate capacity exists for the network resource to handle the request. The Traffic Shaping component Web service will return information about its local token bucket that may be used to drive service implementation queuing.

## Reservation scheme

The Traffic Shaping component Web service implements a distributed token bucket. A token bucket algorithm uses the analogy of a bucket to control the rate of requests. A bucket has a size $B$, which corresponds to the number of tokens that may fit into the bucket. The size of the bucket also corresponds to the maximum burst size. Each request that comes in consumes some tokens from the bucket. When no more tokens are left in the bucket, the request cannot be processed currently and must either be rejected or queued. Tokens regenerate at a rate $R$, which corresponds to the maximum average sustained rate of traffic. In any interval $t$, a maximum of $B + Rt$ tokens may be consumed. This provides shaping of traffic, allowing for bursty output behavior while constraining the average throughput.

The following scheme is used for partitioning network resource traffic among cluster members. The maximum burst size limit is divided up evenly among all active members of the cluster. For example, given a burst limit of 30 tokens and three cluster members, each cluster member will be allocated a 10 token bucket size, or burst size according to the token bucket algorithm. Should one cluster member be offline, the burst limit is repartitioned to the remaining two cluster members for a bucket size of 15 tokens. This conservative scheme ensures that the burst output across the cluster never exceeds the limit. It also works well with round-robin load balancing. A reservation scheme is used to allocate rate amongst members of the cluster to enforce cluster-wide rate limits. A single coordinator is elected among the cluster members. The coordinator maintains reservation information for each cluster member against the cluster-wide limits. Cluster members send reservation requests as needed, that is when insufficient rate is currently allocated, to attempt to allow additional traffic. When the allocated rate is no longer needed (as the bucket begins to fill up faster than tokens are being consumed or becomes full), then a release request is sent to the coordinator to relinquish rate. Rate is released gradually as the bucket level begins to approach the maximum or fully when the bucket reaches its maximum This method accommodates different distributions of burst. A running average of current traffic is maintained and used to determine when rate is no longer needed. Rate is released gradually (every second) in chunks that are a percentage of the maximum sustained average rate.

Reservation requests are made as soon as the token bucket starts to deplete. This tries to regenerate burst in accordance to the current rate of consumption of tokens. The coordinator may deny a reservation request for additional rate if there is not enough remaining rate across the cluster. When a cluster member receives a rejected reservation request, it will enter a silence period. This silence period will suppress additional reservation requests, reducing inter-cluster traffic and allowing the network to settle before attempting additional reservation. In steady state, this minimizes the inter-cluster messaging.

Upon failover of the node housing the cluster coordinator, a new coordinator is elected. A message is resent to all the nodes resetting their reservation status. This results in a ″reset″ of the algorithm without interrupting service. Inter-cluster traffic may increase temporarily while the algorithm rebuilds state to reduce inter-cluster messaging.

## Configuration

Network resources traffic shaping limits are configured in the Network Resources section of the TWSS Administration Console. The Network Resources component Web service must be installed in order for this function to be enabled. A network resource has a logical name that is associated with a resource specification, or a set of properties. Multiple service implementations may be configured to refer to the same network resource logical name.

The Traffic Shaping component Web service expects the following network resource specification properties to be defined:

- **MaxBurstSize**: The maximum amount of burst traffic that can be handled by the network resource. This is measured in number of tokens, where a single service implementation traffic shaping request may consume multiple tokens. The actual constant is 1000.

  **Note:** In most cases it should be lower than 1000.
  This deployment time decision is dependent on the capacity of the backend system.
- **MaxAverageSustainedRate**: The maximum average sustained rate of work, measured in tokens per second, that can be handled by the resource. This corresponds to the rate of token regeneration in the token bucket algorithm.

To define a network resource logical name and its properties, launch the TWSS Administration Console and click **Network ResourcesNetwork Resources Network Resource Names**.

- Click **New**. Then specify a new network resource logical name and provide a description for it.
- Select a network resource logical name in the list. Then click **New** to define its properties and their associated values.

## Guidelines for choosing limits

The maximum burst size must be chosen such that, when burst size is split evenly among all members of the cluster, each member’s local token bucket has sufficient tokens to accommodate the largest request weight. Otherwise, there will never be sufficient tokens to satisfy such a request and this request will always be rejected. In addition, configuring some additional burst above the maximum request size will provide better adaptability of the algorithm to different traffic distributions of traffic being generated within the cluster. An example of such burst is configured a maximum burst of 10 requests, when the maximum request weighting is 1.

When running at capacity, the algorithm has a few for request distributions that approach the traffic rates near the specified limit for the network resource. Reservation requests for rate are made on an as-needed basis when processing incoming requests. Each request may result in a reservation request for enough rate to replenish the tokens consumed by that request. Clusters are typically fronted by a round-robin load-balancer and thus may generate outbound traffic in a similar fashion. When running at capacity, it is possible that an individual node will get allocated a chunk of rate during a round-robin spray and the other remaining members be denied. This may not match expectations across the cluster. For example, consider a cluster with three members, with a rate limit for an operation of 30 tokens per second across the cluster and each cluster member allocated 9 tokens per second. This leaves 3 tokens per second worth of rate to reserve. If an additional spray of requests comes on top of the 27 tokens per second rate across the cluster, then the first request in the spray will result in the first server getting allocated the 3 tokens per second and the remainder being denied additional capacity. This may result in less token regeneration across the cluster than expected. This behavior typically manifests itself as a ping-pong effect, where the last little bit of tokens get traded off between members of the cluster. This can be avoided by running a rate that is less than: (number of cluster members * maximum token per request size). The algorithm will tolerate running within this threshold, but some premature rejections may occur in rare traffic patterns.

## Administering the Network Resources component Web service

The Network Resources component Web service provides a way to define specifications for network resources (elements) with which the service platform will interact. Specifications that you design using the Network Resources component Web service are used to tailor the way in which the service platform behaves toward each resource.

For example, you can use the resource's message processing capacity to control the rate at which the service platform generates traffic toward the element. You can also use specifications to store additional properties regarding protocol interactions with the resource.

This component is used by the Traffic Shaping component Web service to fetch network resource properties that govern the rate of traffic flow through the network.

Resource specifications are flexible so that they can be used by other components and services.

### Configuration

Specifications for network resources are configured within the TWSS Administration Console under the network resources section. A network resource has a logical name that is associated with a resource specification, or a set of properties. Multiple service implementations may be configured to refer to the same network resource logical name. To navigate to the network resource and define the logical name, you will need to do the following:

1. Select TWSS Administration Console **Network ResourcesNetwork ResourcesNetwork Resource Names**.
2. Click **New**. Then define the logical name and its description.

   To define properties for the specific Network resource logical name, do the following:
3. SelectTWSS Administration Console **Network ResourcesNetwork ResourcesNetwork Resource Names**.
4. Under Resource Names, select an existing resource.
5. Click **New**, then add the resource properties and associated values.

The Network Resources component Web service expects the following network resource specification properties to be defined:

- **NetworkResourceNames**: The class name for the list of resources. This dynamic property enables you to create new names for network resource specifications by setting and getting attributes.
- **NetworkResourceAttributeNames**: The list of attributes for the network resource names. This dynamic property enables you to create new properties for network resource specifications by setting and getting attributes.
- **MaximumBurstSize**: The maximum amount of burst traffic that can be handled by the network resource. This is measured in number of tokens, where a single service implementation traffic shaping request may consume multiple tokens.
- **MaximumAverageSustainedRate**: The maximum average sustained rate of work, measured in tokens per second, that can be handled by the resource. This corresponds to the rate of token regeneration in the token bucket algorithm.

The Traffic Shaping component Web service is closely related to this service. Refer to the topic *Administering the Traffic Shaping component Web service* for more information.

### Guidelines for choosing limits

These properties are the network resource attributes that are known for traffic shaping, and are defined by the Web service implementations that use traffic shaping. They are initialized during the system setup when running the DDL files associated with each service implementation. If the administrator decides to tune the system, they can navigate to the network resource component and view the resources that have been defined. The values can be modified from the network resource component.

## Using the Notification Management user interface

The Notification Management user interface provides TWSS administrators with a means to manage the notifications by interfacing with Notification Management Web service client, which would then interface with the Notification Management administration Web service.

The Notification Management user interface is an add-on to the existing notification management common component that provides an easy way for administrators to perform any administrative tasks through a user interface. It is available as an MBean implementation found in the TWSS Administration Console.

## Operation capabilities

The Notification Management user interface supports the following operation capabilities:

- **Listing active notifications**: Displays all the active notifications in the system. Notifications are divided into the number of pages and only ″n″ active notifications are displayed at a time.
- **Listing active notifications by criteria**: Retrieves notification information-based service, requester, or correlator criteria. Notifications are divided into the number of pages and only ″n″ active notifications are displayed at a time.
- **Resetting statistics for all notifications**: Resets the statistics of all the active notifications in the system.
- **Resetting notification statistics by criteria**: Resets notification statistics based on service, requester, or correlator criteria.
- **Terminating active notifications**: Terminates all active notifications from the system.
- **Terminating active notifications by criteria**: Terminates notifications based on service, requester, or correlator criteria.
- **Selective terminate / reset notifications**: Enables the administrators or care representative to reset or terminate the notifications selectively.

## Active Notification Display page

This screen displays the all the active notifications in the system. An active notification has different information fields. The Notification Management user interface is designed to display selected field information about a notification.

The Correlator information is displayed as hyperlink, which if clicked on, displays additional fields on a separate page to provide more information on the active notification.

## Use of filters in the notification display

The notification information can also be filtered by Correlator, ServiceName, and Requester. You can also add filters by clicking on the show/hide filter settings button. Using the filter settings button, it is possible to select or view notifications based on the filter values specified for Correlator, Requester, and ServiceName.

Here are some of the guidelines for using the filters:

- Wild card characters can be used to match one or more similar values.
  - * - Means all / any value
  - ? - Stands for a single character

  Wild card characters can be used in conjunction with any alphanumeric characters.
- Correlator, Requester, and ServiceName are set to * by default.
- Multiple values for Correlator, Requester, and ServiceName should be separated by spaces, for example: corr1 corr2 corr3.
- If there are multiple entries for Correlator, Requester, and ServiceName, such as ([corr1 corr2 corr3], [req1 req2 req3], [serv1 serv2 serv3]) then, (corr1, req1 & serv1) are used to form a unique ID. Similarly, (corr2, serv2, and req3) is used for framing the next unique ID.

- If any of the entries are missing, the last missing entries are assigned to *. For example, consider the following input condition. Input: ([corr1 corr2 corr3], [req1 req2 req3], [serv1 serv3])

  Here (corr1, req1, and serv1) form one unique ID and (corr2, req2, serv3) from another unique ID. (corr3, req3, '*') are used to form the last unique ID.

  **Note:** Even though serv3 is part of the third unique key, it is being used to form the second unique ID.

The following are descriptions of the input fields and action buttons for the filter settings:

- **Correlator**: Specifies the correlator value(s) for the filter conditions. Default '*'
- **Requester**: Specifies the requester value(s) for the filter conditions. Default '*'
- **ServiceName**: Specifies the service name value(s) for the filter conditions. Default '*'
- **Apply to All**: Useful when it is required to terminate or reset statistics of active notifications based on some filter values. Selecting this check box and calling a reset or terminate request would reset or terminate notifications of all the notifications matching the current filter condition. Similarly, filter value '*' in all the three fields with the **Apply to All** check box selected, has same effect as a terminateAll or resetAll Web service request on the Notification Management component.
- **Go**: Updates the current filters settings to the temporary session repository. It also refreshes the active notifications display page with the notifications entries matching the current filter.
-

**Note:** Empty values are not allowed in the filter settings input fields. An error message will be displayed if empty values are used.

## Lifetime of filter settings

Filter settings are not permanently maintained to safeguard against accidental termination of active notifications. All filter settings are cleared in one the following conditions:

- Session timed out
- User-initiated action to clear the filter settings
- User has navigated to another page or revisited the same page following the regular breadcrumb path

## Implementation of Notification Management user interface operations

Notification Management UI supports the Web service operations discussed earlier. The proper use of the filters allows you to perform the various Web service operations. Following describes how to perform different operations using the filter settings.

### List active notifications

By default, the Notification Management user interface displays all the active notifications. It is also necessary to clear the filter settings [if set]. To clear the filter settings, do one of the following:

1. Enable the Filter Settings window by clicking the **Show/Hide Filter** button.
2. Set ‘*’ for all the filter fields.

or

- Click the **Clear Filters** button to clear the filter settings.

#### List active notifications by criteria

Complete the following steps to list active notifications by criteria:

1. Enable the Filter Settings window by clicking the **Show/Hide Filters** button.
2. Enter the appropriate filter value for Correlator, Requester, and ServiceName. Make use of the wild card character ‘*’ wherever necessary.
3. Click **Go** to apply the filters on the result set.

Only the notifications matching the filter condition are now displayed.

#### Terminate all active notifications

Complete the following steps to terminate the all the active notifications:

1. Enable the Filter Settings window by clicking the **Show/Hide Filters** button.
2. Enter the value ‘*’ for Correlator, Requester, and ServiceName.

   **Note:** To terminate the active notifications by criteria, use the wild card character ‘*’ wherever necessary.
3. Select the **Apply To All** check box.
4. Click **Go** to apply the filters on the result set.
5. Click **Terminate** to terminate all the active notifications.

#### Reset statistics of all active notifications

Complete the following steps to reset all the active notifications:

1. Enable the Filter Settings window by clicking the **Show/Hide Filters** button.
2. Enter the value ‘*’ for Correlator, Requester, and ServiceName.

   **Note:** To reset the statistics of all active notifications by criteria, use the wild card character ‘*’ wherever necessary.
3. Select the **Apply To All** check box.
4. Click **Go** to apply the filters on the result set.
5. Click **resetStat** to reset the statistics for all the active notifications.

### Configure the appropriate Notification Management common component

The Notification Management user interface interacts with the Notification Management common component. Therefore, it is necessary to configure the Notification Management component endpoint in the Notification Management user interface settings page.

Service implementations may decide to register the notification with different Notification Management services. In such cases, the Notification Management user interface can only retrieve, reset, or terminate those notifications that are registered with a Notification Management whose service endpoint is configured in the Notification Management user interface settings.

The Web service URL for Notification Management is: http://localhost:9080/TWSS/ServicePlatform/NotificationManagement/services/NotificationAdministration

**Note:** The Notification Management user interface can talk to remote notification managers.

### Configuration properties

The TWSS Administration Console configuration properties are managed through a JMX MBean. When managing the configuration properties, the changes do not require the system to be restarted. Also, MBean configuration settings will beare read at the start of each call, keeping the call settings consistent through the lifecycle of the call.

**Note:** The **MaxNotificationDisplay** setting allows you to configure the number of notifications to be displayed on a page. The default value is 15.

## Administering the PX Notification component Web service

The PX Notification component Web service captures information about activate notifications on the service platform for use by administrators who are monitoring the network.

The PX Notification component Web service delivers the notification, or a Mobile Originated message to the requestor endpoint, through the Access Gateway or directly to the requestor endpoint. The PX Notification component uses the Statistics Publishing Client for publishing delivery statistics in success and failure scenarios. This statistic can be used by the administrator to monitor the traffic from the back-end to the Service Implementation.

The PX Notification component adds a SOAP header called the DeliveryContext, which has information like the requestor ID, the notification destination, and whether ESB should be bypassed. This information is used by the ESB or can be used by the requestor to match the responses with the requests that were sent. The SOAP header can be extracted from the MessageContext for use in a SOAP handler.

### Configuration

Service implementations that use the PX Notification common component may need to ensure that the interface URL endpoints and PORT address pointing to the PXNotification Web service are correct.

All services that use PX Notification also uses the Statistics Publishing Client. The configurable items appear in the Administration Console, with the endpoint, username, and password values defined by your installation. This is accomplished much like any other Service Platform component. The URL defaults to localhost.

**Note:** The PX Notification component needs to be installed on the Service Platform.

### Logging Considerations

The requestor ID is the link between the requests and responses. Service Implementations should include this ID in their logging for tracking purposes. When a Service Implementation uses a Delivery Confirmation Callback endpoint, items such as the global transaction ID, correlation ID, action, failure message, are all sent back to the Service Implementation. This information may also be logged to keep track of delivery statuses of notifications or Mobile Originated messages.

## Administering the Fault and Alarm component Web service

The Fault and Alarm component Web service can be deployed to capture faults and alarms that occur for each deployed Web service implementation, for the Admission Control component Web service, and for the Usage Record component Web service. It can write fault and alarm information to the common event infrastructure (CEI) repository.

If you plan to use the CEI repository to capture fault and alarm data, you must enable the CEI service and configure a data source for it.

The CEI service uses the database included with WebSphere Application Server, but it can be configured to use DB2, Oracle, or another database supported by WebSphere Application Server.

When encountering error conditions, service implementations will need to emit fault information. For severe error conditions, they will need to emit alarms. Both faults and alarms are emitted as common base events using JMX management notifications, and optionally using CEI as well. The common base event is an extensible XML schema for describing event information.

The Fault and Alarm component Web service provides extensions to the common base event format to describe implementation faults. CEI is used as the infrastructure for notifying interested parties of events and provides a persistent event repository that supports flexible event queries based on XPath. JMX notifications are useful for enabling third parties to access WebSphere Application Server and handle notifications, for example by generating SNMP alerts. However, because JMX notifications do not include a listener for CEI, the Fault and Alarm component Web service produces common base events in addition to JMX notifications.

Although CEI can be used to emit any kind of business event, CEI is limited in this implementation to generating fault and alarm events. This is done so that you do not need to set aside a large number of application server instances to process CEI events. The CEI client API comes embedded with the base WebSphere Application Server product and can be activated at no charge.

### Configuration

Settings for faults and alarms are configured within the TWSS Administration Console. The Fault and Alarm component Web service expects the following value to be defined:

- **CEI Notifications Enabled**: A Boolean value indicating whether CEI events are emitted when processing a fault or alarm. If set to `false` (the default), only JMX notifications are emitted. For details, refer to the topic *Enabling the Fault and Alarm component Web service to use CEI*.

To configure the settings for faults and alarms, launch the TWSS Administration Console and click **Web Services Platform Fault Alarm Fault and Alarm Web Service**.

The Traffic Shaping component Web service is closely related to this service. Refer to the topic *Administering the Traffic Shaping component Web service* for more information.

## Disabling Service Platform components for Web services

You can disable one or more Service Platform components if the functionality is not needed in the network.

### About this task

For example, if another mechanism–like the Parlay X Payment (PostPaid) Web service–is collecting charging information, you can disable the Usage Record component Web service.

Similarly, a developer who is testing custom code might want to disable admission control and traffic shaping while testing code.

To disable a Service Platform component, the **End Point URI** field must be empty. If you need to enable a component later, you can do so by typing its endpoint URI into the field.

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services**.
4. Click the name of the Web service implementation for which you want to disable the Service Platform component, for example **IMS Third Party Call**.
5. In the Support Services table, click the name of the Service Platform component you want to disable, for example **Admission Control Client**.
   - To disable the component, delete the URI in the **End Point URI** field.
   - To enable the component, enter its URI in the **End Point URI** field.
6. Click **OK**.

## Administering Web service implementations

Use the Telecom Web Services Server Administration Console to administer the Web service implementations.

To access the configuration for the Web service implementations and Service Platform components, follow these steps:

1. Log in to the Integrated Solutions Console.
2. Open the TWSS Administration Console.
3. In the Navigation pane, click **Web Services**.
4. Click the name of the Web service, for example **IMS Payment** or **IMS Call Notification**.
5. Click the name of the Web service implementation, for example **Payment Web Service** or **Call Notification Web Service**.

# Administering Web service implementations based on SIP/IMS

Telecom Web Services Server supports Web service implementations that operate over SIP/IMS.

## Administering Parlay X Call Notification over SIP/IMS

After you deploy Parlay X Call Notification over SIP/IMS, you can use the TWSS Administration Console to change the deployment configuration properties for the service and related Service Platform components.

Parlay X Call Notification over SIP/IMS provides a default mapping between SIP response codes and Parlay X network event definitions. The administrator can customize this mapping through the TWSS Administration Console. These network events are mainly the response code results from a SIP INVITE request. Response codes range from 1xx to 6xx, with the 4xx to 6xx range being the response codes of interest for this service implementation. For more information about response codes, refer to *RFC 3261*, section 21, *Response Codes*, of the *Network Working Group Request for Comments*.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default runtime values for event notification:
  - **Call Notification Web Service**: This Web service handles calls initiated by a subscriber in the network.
    - **Busy event codes**: The codes that correspond to a Parlay X `Busy` event notification.
    - **No answer event codes**: The codes that correspond to a Parlay X `No Answer` event notification.
    - **Not reachable event codes**: The codes that correspond to a Parlay X `Not Reachable` event notification.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Parlay X Call Notification over SIP/IMS does not use policies.

## Administering Parlay X Presence over SIP/IMS

After you deploy Parlay X Presence over SIP/IMS, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

After completing the installation, configure the SIP URI of Presence Server through the PresenceServerSIPURI setting.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default values used for Parlay X Presence over SIP/IMS.
  - **Additional Group URI Schemes**: This specifies a comma-separated list of URI schemes that the service interprets as indicating a group URI. This setting must be coordinated with the Group List Manager component configuration.
  - **Nested Group Support**: This will indicate that nested groups are supported in group definitions.
  - **Service Implementation Name**: This is used for the Faults and alarms and Usage records.
  - **Default Notification Duration**: The default duration for operations that establish notifications in milliseconds. Smaller values are more restrictive.
  - **Publish Presence Timeout**: The expiration time for Presence information documents which are stored in the Presence server.
  - **Presence Fetch Timeout In Milliseconds**: For SIP-based fetch or CheckImmediate operations, the maximum time a synchronous Web Service should wait for the presence server to send a NOTIFY. When a fetch requires more time, it will receive a NotRetrieved RetrievalStatus notification.
  - **Presence Server Resource Name**: The resource name to use when communicating with the Traffic shaping component.
  - **Inter Operator Identifier**: The InterOperatorIdentifier (ioi), used to create the P-Charging-Vector orig-ioi and term-ioi parameters. If not empty, the service implementation includes P-Charging-Vector headers in its SIP signaling.
  - **Group Support**: Groups that are included with addresses.
  - **Unlimited Count Allowed**: Allowed to specify unlimited notification count, by not specifying the optional count message part in StartNotificationRequest or by specifying a value of zero. False values are more restrictive.
  - **Maximum Notification Frequency**: Maximum rate of notification delivery in milliseconds, which can be considered minimum time between notifications. Smaller values are more restrictive.
  - **Maximum Notification Count**: The maximum number of notifications that may be requested. Smaller values are more restrictive.
  - **Maximum Publish Duration Allowed**: Holds the allowed maximum number in seconds, which is specified for the SIP PUBLISH requests.
  - **Subscribe Presence Timeout**: The maximum lifetime state due to a subscribePresence request, which is specified in milliseconds. Smaller values are more restrictive.
  - **Subscribe Presence WINFO Timeout**: The expiration timeframe for SIP SUBSCRIBE requests.
  - **Presence Server SIP URI**: The SIP URI indicating the location of the backing presence server.

  - **Maximum Notification Duration**: The maximum amount of time that a notification is set up. Smaller values are more restrictive.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from the Service Policy Manager and are passed to the Parlay X Presence over SIP/IMS Web service in the SOAP headers. If the policies are not found in the SOAP header, the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Presence over SIP/IMS*.

## Administering Parlay X Call Handling over SIP/IMS

After you deploy Parlay X Call Notification over SIP/IMS, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default values used for Parlay X Call Notification over SIP/IMS.
  - **Call Handling Web service**: The Parlay X Call Notification over SIP/IMS Web service configuration settings in the Telecom Web Services Server Administration console are as follows:
    - **GroupSupport**: Indicates whether groups may be included with addresses (true) or not (false).
    - **BusyEventCodes**: Comma separated SIP response codes of "Busy" for forward processing. Each code must be three digits and in the range of 400-699.
    - **NoAnswerEventCodes**: Comma separated SIP Response Codes of "NoAnswer" forward processing. Each code must be three digits and in the range of 400-699. These codes cannot overlap with the BusyEventCodes.
    - **RejectResponseCode**: Response code to be sent to the caller when a call is blocked. The value must be in the range of 400-699.
    - **ProvisionalCode**: Provisioning code to be sent to the caller prior to onBusy or onNoAnswer behavior. The value must be in the range of 100-199.
    - **URIParamCheck**: Indicates whether to check if ignored URI parameters are set in the input parameter.
    - **VoiceMailAvailable**: Indicates the availability of voice mail.
    - **TextToSpeechAvailable**: Indicates whether the service accepts text as an input for processing with a Text-To-Speech engine.
    - **AudioContentAvailable**: Indicates whether the service accepts audio content for playing with an audio player.

    - **VoiceXMLAvailable**: Indicates whether the service accepts VoiceXML for processing with a VoiceXML browser.
    - **AudioFormatsSupported**: Indicates the audio formats supported by the service, in a comma separated string format, for example WAV,MP3,AU.
    - **AudioURITemplate**: A parametric template for the SIP URI address of the Media Server for RFC4240 announcement service. The default value is 0.
    - **MediaResourceURITemplate**: A parametric template for the SIP URI address of the Media Server for RFC4240 prompt and collect service. The default value is 0.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X Call Notification over SIP/IMS in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

## Administering Parlay X Third Party Call over SIP/IMS

After you deploy Parlay X Third Party Call over SIP/IMS, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

Configuration changes are read at the start of each call and do not require the system to be restarted. This will keep the call settings consistent through the lifecycle of the call. This information will be stored to the SIP application session.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default values used for Parlay X Third Party Call over SIP/IMS:
  - **Third Party Call Web Service**: For connecting callers. The properties that can be configured include the following.
    - **Media Server URI**: The SIP URI address of the Media server.
    - **Create pHeader**: This is set if the service needs to create a security pHeader.
    - **Service name**: The default service name is PX21_TPC_IMS.
    - **Status retain time**: The amount of time in seconds to keep a call session active after the call terminates. If the value is greater than 59 seconds, the call record is written to the ThirdPartyCall database.
    - **Create charging records**: If present, the Parlay X Third Party Call over SIP/IMS will write these records to the call records.
    - **Traffic estimate per request**: An estimate on how much message traffic will be generated for each request
    - **Timer duration interval**: Listed in seconds. The timer manages the call records in the ThirdPartyCall database and deletes the records that have existed for longer than the status retention time. (This value depends on **Status retention time** having been set such that call records are written to the database.)

    - **Inter Operator Identifier**: This setting is used to define the home originating network identifier; for example: ibm.com
    - **Outbound proxy address**: This setting defines the IMS outgoing proxy when calls are initiated.
    - **Outbound proxy resource specification name**: Describes the needed information for traffic shaping towards the SIP proxy.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from the Service Policy Manager and are passed to the Parlay X Third Party Call over SIP/IMS Web service in the SOAP headers. If the policies are not found in the SOAP header, the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Third Party Call over SIP/IMS*.

# Administering Web service implementations based on IBM WebSphere software

Telecom Web Services Server (TWSS) supports Web service implementations that interoperate with other components within the IBM WebSphere product group–including WebSphere Application Server, IBM XDMS, and other instances of TWSS.

## Administering Parlay X Payment (PostPaid)

After you deploy Parlay X Payment (PostPaid), you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default values used for Parlay X Payment (PostPaid):
  - **Payment Web Service**: For creating payment billing records. The following properties can be configured:
    - **Default for the CEIEnabled property**: Used to enable or disable the default policy for writing payment records to the CEI component.
    - **Default for the currency policy**: Used as the default currency policy value in the payment record if the payment Web service does not specify a currency value in the ChargeInformation structure.
    - **Default for the PaymentUsageRecordsEnabled policy**: Used to enable or disable the default policy for writing payment records to the Usage Record component Web service.
    - **Default for the ResultsUsageRecordsEnabled policy**: Used to enable or disable the default policy for writing payment result records to the Usage Record component Web service.

**Note:** If both the CEIEnabled property and the Payment UsageRecordsEnabled property are set to **false**, a ServiceException will occur, because no payment record will be written to any database or repository.

- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from the Service Policy Manager and are passed to the Parlay X Payment (PostPaid) Web service in the SOAP headers. If the policies are not found in the SOAP header, the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Payment (PostPaid)*.

## Administering Parlay X Address List Manager over XCAP

After you deploy Parlay X Address List Manager over XCAP, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

The Parlay X Address List Manager over XCAP Web service implementation receives Web service inputs from the Access Gateway and interacts with the IBM WebSphere XML Document Management Server (IBM XDMS) component. All operations use the XDM/XCAP interface of IBM XDMS.

Address List Management uses the TWSS Administration Console to configure the definition of constraints of an address book. The service configuration properties for Address List Management are displayed on a **General properties** page in the Console.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- You can change the setting in the TWSS Administration Console **Web servicesIMA Address List ManagerAddress List Manager Web Service**.**Services**: Default values used for Parlay X Address List Manager over XCAP:
  - **UpdateQueryProcessingCostRatio**: The processing cost ratio of updating transactions that query the transactions in the XCAP server. The default is **10**.
  - **MaxGroupMembers**: The number of members in a group against the maximum number that is checked in the XCAP server. If this error occurs, the Address List Manager sends out a POL0210. The default is **100**.
  - **SupportNestedGroups**: Support for nested groups for the criteria of POL0211. The default is **True**.
  - **MaxGroupLength**: The maximum length of the group name for the criteria of POL0212. The default is **20**.
  - **AverageGroupMembers**: The average number of members in a group for workload calculation. The default is **25**.

- **AverageGroupAttributes**: The average number of attributes in a group, or in a member workload calculation. The default is **2**.
- **UseAssertedIdentity**: This is set to *True*, if you use the X 3GPP Asserted Identity. Set to *False*, if you use the HTTP digest authentication. The default is **True**.
- **RootURI**: Used when the Address List Manager generates an XCAP URI to reference a XCAP document within XDMS. It starts with *http://* and ends with */*. For example; `http://fh4a.test.austin.ibm.com:9080/services/`.
- **GroupScheme**: The scheme name of a group used for generating the actual group URI. The value should be same as groupScheme of XDMS MBean. The default is **Group**.
- **UserScheme**: A scheme name for the users. The default is **sip**.
- **ProviderDomain**: The provider domain name used for generating the actual group URI.
- **DefaultUserFileName**: A file name for users in the XCAP URI when the null string is specified in the domain part of the *createGroup* operation. The default is **user_file**.
- **DefaultGlobalFileName**: The file name for the global in XCAP URI, when the null string is specified in the domain part of the *createGroup* operation. The default is **global_file**.

### Connecting to IBM XDMS

To connect Address List Management to an IBM XDMS server, specify the root URI as `http://hostname:port/services`, where *hostname* and *port* are the appropriate values for the IBM XDMS server. Refer to the topic *Finding the IBM XDMS XCAP root* for help with locating these values.

### XCAP passwords

If the XCAP server requires passwords, you can configure them by defining a policy as follows:

1. Launch Service Policy Manager console.

   **Note:** If you are using the Service Policy Manager console with the Firefox browser, Firefox version 3.0 is strongly recommended.
2. Set the password policy for each requester, using **service.config.auth.Password** in the Service Policy Manager Console. The policy type should be ″**Password**,″ and this will encrypt the policy.

### Policies

Parlay X Address List Manager over XCAP does not use policies.

## Administering Web service implementations based on Direct Connect protocols

Telecom Web Services Server supports Web service implementations that operate over Direct Connect protocols such as SMPP, MLP, and MM7.

### Administering WAP Push over SMPP

After you deploy WAP Push over SMPP, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

You can receive additional information on specific properties from the online help in the TWSS Administration Console. Click **More information about this page** at the top right corner of the panel. Help for the module is then displayed in a separate browser window.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Global**: Policy values that are used for all services.
  - **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
    - **Country and code**: The local country and code.
    - **Address plan**: The local address plan.
    - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
    - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

    For more information about these settings, refer to the topic *Group schema configuration*.
- **Services**: Default values used for WAP Push over SMPP:
  - **WAPPush Service**: The following properties can be configured:
    - **Purge Enabled**: True if cleanup activity occurs for expired delivery status data; false if cleanup activity does not occur. The default value is `false`.
    - **Purge Interval**: The time interval, in minutes, used to trigger the cleanup operation for expired delivery status data.
    - **Purge Block**: The number of records to retrieve each time there is a query to find expired data records. In effect, the cleanup operation is limited to cleaning up this much data during each purge interval. A value of 0 indicates that all records found should be removed. The default value is 1000.
    - **Service Enabled**: True if the WAP Push over SMPP service is enabled; false if it is not enabled. The default value is `true`.
    - **Network Resource Name**: The name of the network resource. Must match the name specified in **Network Resources** → **Network Resources Configuration** → *resource_name* for WAP Push over SMPP. The default value is `WAP10_PUSH_SMPP`.
- **Network Resources**: Configure network elements to prepare them to receive messages from WAP Push over SMPP.
  - **WAP Push Backends**: You can configure the following properties for the WAP Push SMPP Network Mapper:
    - **Backend Name**: The name of the backend server that will communicate with the SMSC.
    - **Hostname**: The host name for the target SMSC.
    - **Port**: The port of the target SMSC. The default value is 2775.
    - **System ID**: The System ID that is used to bind to the SMSC.
    - **Password**: The password for the System ID.

  - **System Type**: The type of ESMC system that is requesting to bind as a transceiver with the SMSC. The input string must be 12 characters or less. The default value is a null string.
  - **Max Message Size**: The maximum allowable size, in bytes, of an individual message as stipulated by the SMSC. When an incoming message is larger than MaxMessageSize, it will be split into multiple messages (segmented) before sending. If the incoming message is larger than the maximum allowed by the protocol but smaller than MaxMessageSize, the message will be sent using the SMPP message_payload protocol. Valid values are 0 through 64000. The default value is 245. When the value is 0, no segmentation will occur.
  - **Max Targets Size**: The maximum number of targets allowed in a single message, according to the protocol specification or network support. Valid values are 0 through 255. The default value is 255. When the value is 0, no maximum number is enforced.
  - **Confirmed Delivery**: True if a delivery receipt will be requested on each message sent to the SMSC; false if not. The default value is `false`.
  - **Data Coding**: The type of encoding (DataCoding or DCS value) used for messages on the target SMSC. There are two possible values for WAP messages: 0x04 and 0x05.

    To set DataCoding to 0x04, select `8_bit_binary_GSM_23_038_a`.

    To set DataCoding to 0x05, select any other value.

    The default value is `ISO-8859-1`.
  - **Type of Bind to SMSC**: The bind type to send to the SMSC. TWSS supports three types of bindings, which are defined in the *SMPP Protocol Specification*. The field is a drop-down list containing the following choices:
    - Transceiver(Trx): The connection can be used both for sending Mobile Terminated (MT) messages to the SMSC and for receiving Mobile Originated (MO) messages from the SMSC.
    - Transmitter(Tx): The backend will be used only for sending Mobile Terminated (MT) messages. Any Mobile Originated (MO) messages sent from the SMSC using the connection will be discarded.
    - Receiver(Rx): The backend will not be used for sending Mobile Terminated (MT) messages. It will be used only for receiving Mobile Originated (MO) message sent from the SMSC. Note that outbound ENQUIRE_LINK operation are not initiated on back ends that use Rx mode binding.

    The default value is Transceiver(Trx).
  - As in TWSS 7.0, continue to provide support for the message types Synchronous SMS, Statusless SMS, and Asynchronous SMS. In the SMS SI WS, the existing MBean attributes, namely **Enable Synchronous SMS** and **Enable StatusLess SMS**, in the SMS backend configuration page, help to configure the message type. These are merged as a single MBean attribute named “MessageType,” which has the values **SynchronousSMS**, **StatusLessSMS**, and **AsynchronousSMS**.

    **Note:** If set to `true`, you will need to configure connection pool settings at the following location in the Integrated Solutions Console: **Resource adapters** → **SMPP v3.4 JCA Connector** → **J2C connection factories** → **PX21_SMS_SMPPJ2CCF** → **Connection pools**. The default value is `false`.
- **WAP Push Alias Details**: The following properties can be configured:

        - **Alias**: Defines a mapping between the backend server details and a range of target addresses configured as SMS service policies. For every alias, at least one of the backends must be configured with a valid value.
        - **Primary Backend Server**: The primary backend server for handling messages.
        - **Secondary Backend Server**: The secondary backend server for handling messages, used when the primary server is not available.
    - **WAPPush SMPP ENQUIRELINK and Bind Settings**: The following properties can be configured:
        - **Enable Enquire Link Service**: True if the enquire link service is enabled, false if disabled. The default value is `false`.
        - **Enquire Link Interval**: The time interval, in minutes, to be used between enquire link calls. Valid values are 1 through 60. The default value is 5.
        - **service_type Usage**: Specifies the value of the **SMPP PDU** field used for sending charging information to a backend server or used to identify the type of service in the delivery response. The field is a drop-down list containing the following choices:

            **For WAP and SMS over SMPP co-deployment**: Use if you are deploying the WAP Push Web service together with SMS over SMPP on the same server.

            **For passing charging information**: Use if you are *not* deploying the two services together. (This is the default value.)
        - **service_type Value**: Specifies the value of the **service_type** field to be set in the SMPP PDU that is sent to a backend server or included in the delivery response.
            - If **service_type Usage** is set to **For WAP and SMS over SMPP co-deployment**, then **service_type Value** is used for sending charging information to the SMSC. It is a text string of up to five characters. The default value is `WAP`. In addition to setting **service_type Value**, use the service.config.messaging.Billing policy to identify the desired charging code. The policy value is used to populate the **billing_identification** field in the PDU.
            - If **service_type Usage** is set to **For passing charging information**, then **service_type Value** is not required. Instead, define the service.custom.messaging.ServiceType policy. The policy value is used to populate the **service_type** field in the PDU. Note that in this case you cannot deploy the WAP Push Web service together with SMS over SMPP on the same server.
    - **WAPPush SMPP ESMC Settings**:

        The SMPP-based Web service implementations (SMS over SMPP and WAP Push over SMPP), use predefined values when communicating with the Short Message Service Center (SMSC). The values are used in each message sent to the SMSC. You can modify these values using the TWSS Administration Console.

        For more information, refer to the topic *Configuring communication with the SMSC*.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from the Service Policy Manager and are passed to the WAP Push over SMPP Web service in the SOAP headers. If the policies are not found in the SOAP header, the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for WAP Push over SMPP*.

## Administering Parlay X SMS over SMPP

After you deploy Parlay X SMS over SMPP, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

You can receive additional information on specific properties from the online help in the TWSS Administration Console. Click **More information about this page** at the top right corner of the panel. Help for the module is then displayed in a separate browser window.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Global**: Policy values that are used for all services.
  - **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
    - **Country and code**: The local country and code.
    - **Address plan**: The local address plan.
    - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
    - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

    For more information about these settings, refer to the topic *Group schema configuration*.
- **Services**: Default values used for Parlay X SMS over SMPP:
  - **SMS Service**: The following lists the configurable runtime properties:
    - **Purge Enabled**: True if cleanup activity occurs for expired delivery status data; false if cleanup activity does not occur. The default value is `false`.
    - **Purge Interval**: The time interval, in minutes, used to trigger the cleanup activity on Parlay X SMS over SMPP data. The default value is 10.
    - **Purge Block**: The number of records to retrieve each time there is a query to find expired data records. In effect, the cleanup operation is limited to cleaning up this much data during each purge interval. A value of 0 indicates that all records found should be removed. The default value is 1000.
    - **Service Enabled**: True if the service is enabled, false if disabled (when false, all requests are rejected). The default value is `true`.

        - **Network Resource Name**: The name of the network resource. Must match the name specified in **Network Resources** → **Network Resources Configuration** → *resource_name* for Parlay X SMS over SMPP. The default value is `PX21_SMS_SERVICE`.
        - **Default Requester for Orphan Messages**: The requester name to be included in orphan messages. In a notifySmsReception operation, an orphan message is defined as a message that is sent to a target where notification is not enabled, or a message in which the SMS destination does not fulfill the criteria specified in a previous startSmsNotification operation.
        - **Default Endpoint for Orphan Messages**: Endpoint to be notified when an orphan message is received.
        - **Default Correlator for Orphan Messages**: Correlator for the incoming orphan MO messages, used to correlate the messages while retrieving the received MO messages. The default value is an empty string, which indicates that any criteria will be matched. If you specify a non-empty string, then a given message will be retrieved only when a matching correlator is provided.
        - **Default Expiry Time for Orphan messages**: Amount of time, in minutes, to keep the orphan MO messages that have been received in the database. The default value is 1440.
- **Network Resources**: Configure network elements to prepare them to receive messages from Parlay X SMS over SMPP.
    - **SMS Backends**: The SMPP network mapper provides configuration for binding to the backend SMSC.
        - **Backend Name**: The name of the backend that will communicate with the SMSC.
        - **Hostname**: The location of the target SMSC.
        - **Port**: The port of the target SMSC.
        - **System ID**: The System ID that is used to bind to the SMSC.
        - **Password**: The password for the System ID.
        - **System Type**: The type of ESME system that requests the connection to the SMSC. The input string must be 12 characters or less. The default value is a null string.
        - **Max Message Size**: The maximum allowable size, in bytes, of an individual message as stipulated by the SMSC. If the incoming message size is greater than MaxMessageSize, it will be sent using the SAR message segmentation policy. If the incoming message size is greater than the protocol maximum but smaller than MaxMessageSize, the message will be sent using the message_payload. Valid values are 1 through 64000. The default value is 260.
        - **MaxTargetsSize**: The maximum number of targets allowed in a single message according to the protocol specification or network support. Valid values are 0 through 255. A value of 0, which is the default, indicates that there is no limit.
        - **Confirmed Delivery**: True if a delivery receipt will be requested on each message sent to the SMSC; false if not. The default value is `true`.
        - **Data Coding**: The type of data coding to use for the messages on the target SMSC. The default value is `US-ASCII`.
        - **Bind Mode**: The bind type to send to the SMSC. TWSS supports three types of bindings, which are defined in the *SMPP Protocol Specification*. The field is a drop-down list containing the following choices:

      - Transceiver(Trx): The connection can be used both for sending Mobile Terminated (MT) messages to the SMSC and for receiving Mobile Originated (MO) messages from the SMSC.
      - Transmitter(Tx): The backend will be used only for sending Mobile Terminated (MT) messages. Any Mobile Originated (MO) messages sent from the SMSC using the connection will be discarded.
      - Receiver(Rx): The backend will not be used for sending Mobile Terminated (MT) messages. It will be used only for receiving Mobile Originated (MO) message sent from the SMSC. Note that outbound ENQUIRE_LINK operation are not initiated on back ends that use Rx mode binding.

      The default value is Transceiver(Trx).
    - As in TWSS 7.0, continue to provide support for the message types Synchronous SMS, Statusless SMS, and Asynchronous SMS. In the SMS SI WS, the existing MBean attributes, namely **Enable Synchronous SMS** and **Enable StatusLess SMS**, in the SMS backend configuration page, help to configure the message type. These are merged as a single MBean attribute named "MessageType," which has the values **SynchronousSMS**, **StatusLessSMS**, and **AsynchronousSMS**.

      **Note:** If set to `true`, you will need to configure connection pool settings at the following location in the Integrated Solutions Console: **Resource adapters** → **SMPP v3.4 JCA Connector** → **J2C connection factories** → **PX21_SMS_SMPPJ2CCF** → **Connection pools**. The default value is `false`.
  - **SMS Alias Details**: The alias represents a mapping between the target address for a short message and the back end network element that will receive the physical message.
    - **Alias**: The alias to assign the given range of addresses.
    - **Primary Backend Server**: Primary server for handling messages.
    - **Secondary Backend Server**: Secondary server for handling messages. If the primary server is not available, the secondary server is used.
  - **SMS SMPP ENQUIRELINK and Bind Settings**: The following properties can be configured:
    - **Enable Enquire Link Service**: True if the enquire link service is enabled, false if disabled. The default value is `false`.
    - **Enquire Link Interval**: The time interval, in minutes, to be used between enquire link calls. Valid values are 1 through 60. The default value is 5.
    - **Connection Retry Interval**: Time interval, in minutes, between connection retries. Valid values are 3 through 60. The default value is 3.
    - **service_type Usage**: Specifies the value of the **SMPP PDU** field used for sending charging information to a backend server or used to identify the type of service in the delivery response. The field is a drop-down list containing the following choices:

      **For WAP and SMS over SMPP co-deployment**: Use if you are deploying the SMS Web service together with WAP Push on the same server.

      **For passing charging information**: Use if you are *not* deploying the two services together. (This is the default value.)
    - **service_type Value**: Specifies the value of the **service_type** field to be set in the SMPP PDU that is sent to a backend server or included in the delivery response.

  - If **service_type Usage** is set to **For WAP and SMS over SMPP co-deployment**, then **service_type Value** is used for sending charging information to the SMSC. It is a text string of up to five characters. The default value is `SMS`. In addition to setting **service_type Value**, use the service.config.messaging.Billing policy to identify the desired charging code. The policy value is used to populate the **billing_identification** field in the PDU.
  - If **service_type Usage** is set to **For passing charging information**, then **service_type Value** is not required. Instead, define the service.custom.messaging.ServiceType policy. The policy value is used to populate the **service_type** field in the PDU. Note that in this case you cannot deploy the WAP Push Web service together with SMS over SMPP on the same server.

  – **SMS SMPP ESMC Settings**:

  The SMPP-based Web service implementations (SMS over SMPP and WAP Push over SMPP), use predefined values when communicating with the Short Message Service Center (SMSC). The values are used in each message sent to the SMSC. You can modify these values using the TWSS Administration Console.

  For more information, refer to the topic *Configuring communication with the SMSC*.

- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X SMS over SMPP Web service in the SOAP headers. If the policies are not found in the Telecom Web Services Server SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X SMS over SMPP*.

## Administering Parlay X Terminal Location over MLP

After you deploy Parlay X Terminal Location over MLP, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

Parlay X Terminal Location over MLP enables applications to send Web service requests to TWSS, requesting the Terminal Location services defined by the Parlay X 2.1 specification and register for Terminal Location Notifications. The Terminal Location over MLP 3.1/3.2 components are location clients to the location servers.

The component provides the Parlay X 2.1 based Web service that accepts requests for Terminal Location services, controls validity checks that are done to ensure the requests are valid, and forwards the requests to the MLP connector. This is based on MLP 3.1 and MLP 3.2 specification for downstream processing.

## TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Global**: Policy values that are used for all services.
  - **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
    - **Country and code**: The local country and code.
    - **Address plan**: The local address plan.
    - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
    - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

    For more information about these settings, refer to the topic *Group schema configuration*.
- **Services**: Default values used for Parlay X Terminal Location over MLP:
  - **Terminal Location Web service**: The following lists the configurable runtime properties:
    - **Location Type**: Type of location requested, either `CURRENT`, `LAST`, `CURRENT_OR_LAST`, or `INITIAL` (default `CURRENT`).
    - **Altitude Requested**: If true, then altitude information should be requested when location requests are made.
    - **Request Priority**: The priority of the location request, either `HIGH` or `NORMAL` (default `NORMAL`).
    - **Minimum Acceptable Accuracy**: The minimum possible value for acceptable accuracy, expressed as a radius in meters (default 100).
    - **Minimum Requested Accuracy**: The minimum possible value for requested accuracy, expressed as a radius in meters (default 100).
    - **Maximum Age**: Maximum acceptable age, in seconds, of the location information that is returned (default 0, meaning no maximum).
    - **Response Timeout**: Time, in seconds, to wait for the response from the location server, used as a backup for the response time parameter in location-based operations (default 60). A timeout value of zero is interpreted as an infinite timeout. This value is used when no socket timeout is set in the HTTP method parameters.
    - **Use Extended Location**: If true, extended location requests, containing target information and requesting a particular quality of service, are made (default `false`).
    - **Maximum Location Addresses**: The maximum number of addresses for which a `getlocation` for group request can be made (default 5).
    - **Service ID**: A unique ID assigned to the TWSS TL MLP SI by the MLP Location Server. When the value is a non-empty string, it is mapped to the element “client/serviceid” in the MLP Request XML. Otherwise, “client/serviceid” is not added to MLP Request XML.
    - **Requestor Required**: Determines whether to add a requestor element to the MLP Request XML. When true, the WebSphere Application Server User

    Principle is mapped to the element “requestor/id” in the MLP Request XML. When false, the “requestor/id” is not included in the MLP Request XML.
    - **Request Mode**: Type of the request to be sent to MLP Location Server. When it is NONE, the “client/requestmode@type” element is not included in the MLP Request XML. When it is ACTIVE/PASSIVE, the “client/requestmode@type” is mapped to ACTIVE/PASSIVE.
  - **Terminal Location Notification Web service**: The following lists the configurable runtime properties:
    - **Maximum Trigger Radius**: The largest possible notification radius, in meters (default 50000).
    - **Minimum Trigger Radius**: The smallest possible notification radius, in meters (default 100).
    - **Unlimited Notification Count Allowed**: If true, then clients may request an unlimited number of notifications (default `false`).
    - **Maximum Notifications Count**: The maximum number of notifications that can be requested (default 10).
    - **Maximum Notification Duration**: The maximum duration, in milliseconds, of a notification request (default 100000, or 100 seconds).
    - **Maximum Notification Addresses**: The maximum number of addresses for which a notification can be set up (default 10).
    - **Maximum Notification Frequency**: The minimum time, in milliseconds, between notifications of a notification request (default 0).
    - **Minimum Tracking Accuracy**: The minimum possible value for tracking accuracy, expressed as a radius in meter and used in Geographical Notification requests (default 100).
    - **Minimum Requested Accuracy**: The minimum possible value for requested accuracy, expressed as a radius in meter and used in Periodic Notification requests (default 100).
    - **Ignore Check Immediate**: If false, then the terminal’s location is checked while setting up the geographical notifications (default `true`).
    - **CallbackEndpoint**: A virtual callback endpoint URL, where the listener is running or a firewall is configured which routes the request. Example: `http://localhost:9080/TWSS/ParlayX21/TerminalLocation/MLP/services/TerminalLocation`
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

You can receive additional information on specific properties from the online help in the TWSS Administration Console.

## Policies

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Terminal Location over MLP Web service implementation in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service implementation uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Terminal Location over MLP*.

### URI formats

The Parlay X Terminal Location over MLP Web service implementation does not provide support for SIP URIs that include the context.

In the Parlay X Terminal Location over MLP Web service implementation for MLP 3.1, only the following URI format is supported:`sip:+358-555-1234567;postd=pp22@example.com;user=phone`

In the Parlay X Terminal Location over MLP Web service implementation for MLP 3.1, the following URI format is not supported:`sip:+358-555-1234567;phone-context=5;tsp=a.b@example.com;user=phone`

### Altitude information

MLP provides the `alt_acc` (accuracy of the altitude returned) tag for SLIR/TLRR requests. For a potential altitude value, the parameter `alt` is used.

The response handles the altitude information as follows:
- If the extended location request is being opted (where altitude is also expected) and if the response does not contain altitude information, then 0.0 is added for altitude in the response.
- If the extended location request is being opted and the response does contain altitude information, then the actual altitude value is added in the response.
- If the extended location is not being opted and the MLP Location server sends the altitude information, then the altitude value is not considered for the response.
- If the extended location request is not being opted and the response does not contain altitude information, then the response contains no value for altitude.

## Administering Parlay X Multimedia Messaging over MM7

After you deploy the Parlay X Multimedia Messaging over MM7 Web service, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

Parlay X Multimedia Messaging over MM7 is a service implementation over an MM7 interface protocol, which is used to communicate with network elements. It depends on the Access Gateway to insert a transaction identifier into the original Web service request that is received by the Web Service Controller in the MMS service.

The Web service's deployment descriptor, located in the Web Service Controller layer, includes entries for the handlers that are used for each Web service operation.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:
- **Global**: Policy values that are used for all services.

- **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
  - **Country and code**: The local country and code.
  - **Address plan**: The local address plan.
  - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
  - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

  For more information about these settings, refer to the topic *Group schema configuration*.

- **Services**: Default values used for Parlay X Multimedia Messaging over MM7:
  - **Multimedia Messaging Web Service**: The following properties can be configured:
    - **Purge Enabled**: Specifies whether cleanup activity occurs (`true`) or does not occur (`false`) for expired delivery status data. The default value is `false`.
    - **Purge Interval**: Time interval, in minutes, used to trigger the cleanup operation for expired delivery status data.
    - **Purge Block**: Number of records to retrieve each time there is a query to find expired data records. In effect, the cleanup operation is limited to cleaning up this much data during each purge interval.
    - **Service Enabled**: Specifies whether the Parlay X Multimedia Messaging over MM7 service is enabled (`true`) or disabled (`false`). The default value is `true`.
    - **Network Resource Name**: The name of the network resource. Must match the name specified in **Network Resources** → **Network Resources Configuration** → **Resource Name** for Parlay X Multimedia Messaging over MM7. The default value is `PX21_MM_MM7`.
    - **Destination Element**: Defines the element to be used to get the destination number. This configuration option is used to switch between the `<Sender>` and `<Recipient>` elements of DeliveryReportReq, depending on the MMSC's interpretation of the MM7 spec. The default value is `Recipient`.
    - **InMemoryAttachmentHandling**: Determines whether the MMS attachments should be persisted into the database or not, for the sendMessage operation. Setting the value to `true` ensures the attachments are handled in-memory and the operation is performed in a synchronous manner. The default value is `false`.
  - **Multimedia Messaging HTTP Connector Settings**: HTTP default connection settings that are used to send MMS messages to the MMSC. These settings are commonly applicable to all the backends that are created under the **MMS Backends** option. The following properties can be configured:
    - **Default connections per host**: The maximum number of connections that can be configured for any host configuration. The default value is `20`.
    - **Maximum total connections**: The maximum number of connections that can be configured for all hosts. The default value is `100`.
    - **Connection time out period**: The time, in milliseconds, to wait before establishing a connection. A value of 0 means that the timeout is not used. The default value is `0`.

    - **Socket time out period**: The socket timeout time, in milliseconds. This value represents the timeout for waiting for data. A value of 0 means an infinite timeout. The default value is `0`.
    - **Enable MMS SI to make security enabled calls to MMSC**: `true` if security is enabled for the MMS service implementation (SI) to make HTTP calls to the MMSC server using the user name and password provided; `false` if security is not enabled. When security is not enabled, the user name and password are not used to make the connection. The default value is `false`.
    - **Gateway Support for MMS Message without Attachment**: Defines the gateway support for MMS messages without attachments: if `true`, allow the MMS service indication (SI) to send MMS message without attachments to the gateway; if `false`, generate an exception when an attempt is made to send an MMS message without attachments. The default value is `true`.
- **Network Resources**: Configure network elements to prepare them to receive MMS messages.
  - **Multimedia Messaging Backends**: The following properties can be configured:
    - **Backend**: The name of the backend currently being edited.
    - **Host Name**: The host name or IP address of the target MMSC server. The default value is `localhost`.
    - **Port**: The port for the target MMSC server. The default value is `7000`.
    - **URL Path**: The URL path of the target MMSC server.
    - **System ID**: The user name or ID for binding to the MMSC. The default value is `/MMSCSimulator/MMSCServlet`.
    - **Password**: The password for binding to the MMSC.
    - **Maximum Message Size**: The maximum number of characters in messages that are sent to the MMSC. When a message is larger, it will be split into multiple messages (segmented) before sending. The default value is `260`.
    - **Maximum Targets Size**: The maximum number of targets allowed in a single message. The default value is `255`.
    - **Confirmed Delivery**: Specifies whether a delivery receipt is requested (`true`) or not requested (`false`) for each message sent to the MMSC. The default value is `false`.
    - **Data Coding**: Indicates the data coding type to be used for the messages on the target ESMC. The default value is `MMSC Default Encoding`.
  - **Multimedia Messaging Alias Details**: The following properties can be configured:
    - **Primary Backend Server**: The name of the primary server for handling messages.
    - **Secondary Backend Server**: The name of the secondary server, used if the primary server is unavailable.
  - **Multimedia Messaging Optional Parameters Settings**: The following properties can be configured:
    - **VASP ID**: The Value Added Service Provider (VASP) ID configured at the messaging gateway.
    - **VAS ID**: Identifies the originating application.
    - **Linked ID**: Identifies a correspondence to a previous valid message delivered to the VASP.
    - **Message Class**: The class of the MMS message.

    - **Charged Party**: Identifies which party is expected to be charged for a message submitted by the VASP.
    - **Allow Adaptations**: Indicates whether the VASP allows adaptation of the message content (`true`) or not (`false`). The default value is `true`.
    - **Distribution Indicator**: Indicates whether the VASP allows the message content to be redistributed (`true`) or not (`false`). The default value is `false`.
  - **Mobile Originated (MO) message end point configuration for MMS/MM7**: Mobile Originated messages are sent by MMSC to the MMS SI. You have to configure the MMSC URI below, in order for the MO messages to be sent successfully to MMS.

  ```
  http://<Host_Name>:<Port>/TWSS/ParlayX21/MM7Receive/MultimediaMessaging/
  MM7
  ```

  - **Host_Name**: This is the fully qualified host name of the server where the MMS/MM7 application is running.
  - **Port**: The port used to access MMS/MM7 SI.
  - **Context_Root=/TWSS/ParlayX21/MM7Receive/MultimediaMessaging/MM7**: This is the MO receive servlet context root where the MMS SI servlet will be listening for the incoming MO messages.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### MMS/MM7 service RFC2822 support

MT operations for the RFC2822 email address can be performed using one of the following formats:
- sip:+E.164/TYPE=PLMN@recipient-mmse
- sip:+E.164@recipient-mmse
- E.164@recipient-mmse

where “sip” is the schema part that identifies the email address format, “E.164” is the actual MSISDN number, and “@recipient-mmse” is the email address domain.

Example:
- sip:9886061490@in.ibm.com
- sip:+9886061490@in.ibm.com
- sip:+9886061490/TYPE=PLMN@in.ibm.com

Per the MM7 specification, the incoming MOs RFC2822 address should adhere to the following format:

+E.164/TYPE=PLMN@recipient-mmse

Example: +358401234567/TYPE=PLMN@mmse.sonera.net

### Policies

Policy attributes are retrieved from the Service Policy Manager and are passed to the Parlay X Multimedia Messaging over MM7 Web service in the SOAP headers.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Multimedia Messaging over MM7*.

# Administering Parlay-based Web service implementations

Telecom Web Services Server supports Web service implementations that operate over Parlay 3.x and 4.x APIs with Parlay 5.0 Multimedia Messaging (MMM).

## Administering Parlay X Terminal Status over Parlay

After you deploy Parlay X Terminal Status over Parlay, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

The Parlay X Terminal Status over Parlay Web service describes how each individual Web service operation is to be mapped to Parlay Mobility Service Capability Feature (SCF) API methods. Parlay X Terminal Status over Parlay enables applications to send status and notification requests for a terminal, as defined by the Parlay X 2.1 specification.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Global**: Policy values that are used for all services.
  - **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
    - **Country and code**: The local country and code.
    - **Address plan**: The local address plan.
    - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
    - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

    For more information about these settings, refer to the topic *Group schema configuration*.
- **Services**: Default values used for Parlay X Terminal Status over Parlay.
  - **Terminal Status Parlay**
    - **Network Resource Name**: The name of the network resource. Must match the name specified in **Network Resources** → **Network Resources Configuration** → **Resource Name** for Parlay X Terminal Status over Parlay. The default value is `PX21_TS_Parlay`.
    - **Parlay Service Name**: Name of the Parlay gateway service.
    - **Response Timeout**: Indicates the amount of time in seconds to wait for the service activity. The service execution must complete within this window, or a `ServiceException` is returned to the client.
    - **Parlay Automatic Retry**: Indicates whether a Parlay error should cause the service to retry an operation. This is **true** if multiple gateways are configured for a failover.
    - **Number of Parlay Automatic Retries**: The number of times the service will retry a Parlay operation.
    - **Default value for Terminal Status over Parlay**: PX21_TS_Parlay

  - **Terminal Status Notification Parlay**
    - **Ignore Check Immediate**: Indicates whether the `checkImmediate` flag should be ignored because the Parlay gateway automatically checks device attributes when a notification is initially created.
  - **Terminal Status Controller**
    - **Terminal Status Enabled**: Specifies whether the Parlay X Terminal Status over Parlay service is enabled (`true`) or disabled (`false`).
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Service Polices are configured through the Service Policy Manager. These policies are expected to be in the SOAP header. However, they may be stored elsewhere. The Common layer must extract and provide the actual policy values.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Terminal Status over Parlay*.

## Administering Parlay X Terminal Location over Parlay

After you deploy Parlay X Terminal Location over Parlay, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

Terminal Location enables applications to send Web services requests requesting the Terminal Location services defined by the Parlay X 2.1 specification and to register for Terminal Location notifications. Terminal Location offers a simple Web services API that adapts the Parlay X 2.1 API to the Parlay API. The Parlay Connector is responsible for making and managing this connection, as well as CORBA calls to the database.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Global**: Policy values that are used for all services.
  - **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
    - **Country and code**: The local country and code.
    - **Address plan**: The local address plan.
    - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
    - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

    For more information about these settings, refer to the topic *Group schema configuration*.
- **Services**: Default values used for Parlay X Terminal Location over Parlay:

- **Terminal Location Web service**: The following lists the configurable runtime properties:
  - **Network Resource Name**: This field defines the Network Resource name, this name should be same as the one defined under **Network Resources** -> **Network Resources Configuration** -> **Resource Name** for Parlay X Terminal Location over Parlay. The default value is `PX21_TL_Parlay`.
  - **Periodic Notification Supported**: Indicates the availability of periodic notification.
  - **Geographical Notification Supported**: Indicates whether notifications may be set on a geographical area.
  - **Terminal Location Enabled**: True if the service is enabled, false if disabled and all requests will be rejected.
- **Terminal Location Parlay**:
  - **Parlay Service Name**: Parlay gateway service name.
  - **Minimum Triggered Notification Radius**: The minimum notification radius allowed, in meters.
  - **Ignore Check Immediate**: Indicates whether the checkImmediate flag should be ignored because the Parlay gateway automatically checks device attributes when a notification is initially created.
  - **Parlay Automatic Retry**: Indicates whether a Parlay error should cause the service to retry an operation. Should be true if multiple gateways are configured for failover.
  - **Use Extended Location**: Determines whether extended location requests should be made.
  - **Parlay Gateway Location Method List**: List of Parlay gateway location method strings. Entries in the list must match a location method supported by the Parlay gateway. See LocationAccuracyRangeList and the Parlay 4.x Mobility Specification. (This is ignored if `UseExtentedLocation=false`.)
  - **Use Degrees for Notification Radius**: Indicates whether the gateway expects major and minor axis measurements in degrees.
  - **Gateway Reporting Interval Precision**: Units of time passed to the Parlay gateway, in seconds or milliseconds.
  - **Parlay Gateway Location Accuracy Range List**: Integer list of location ranges. Entries are used to select the corresponding location method and parameters for Parlay 4.x Mobility requests. Entries must be in ascending order. (This is ignored if `UseExtentedLocation = false`)
  - **Parlay Gateway Location Type List**: List of Parlay gateway location type strings, which must consist of the following strings:

    ```
    P_M_CURRENT_OR_LAST_KNOWN
    P_M_CURRENT
    P_M_INITIAL
    ```

    See *LocationAccuracyRangeList* and the Parlay 4.x Mobility Specification.
  - **Use Camel Interface**: Indicates whether the CAMEL (Customized Applications for Mobile Network Enhanced Logic) interface should be used to make Parlay requests.
  - **Maximum Triggered Notification Radius**: The maximum notification radius allowed, in meters.
  - **Number of Parlay Automatic Retries**: The number of times the service will retry a Parlay operation.

    - **Response Timeout**: Indicates the amount of time, in seconds, to wait for service activity. Service execution must complete within this window or a ServiceException will be returned to the client.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

You can receive additional information on specific properties from the online help in the TWSS Administration Console.

### Policies

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Terminal Location over Parlay Web service in the SOAP headers. If the policies are not found in the SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Terminal Location over Parlay*.

## Administering Parlay X SMS over Parlay

After you deploy Parlay X SMS over Parlay, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Global**: Policy values that are used for all services.
  - **Common Service Settings**: Use these settings to configure naming schemes for groups. The Access Gateway uses the IBM XDMS component to resolve groups, and it can define and configure groups apart from the group scheme.
    - **Country and code**: The local country and code.
    - **Address plan**: The local address plan.
    - **Enable Transaction Monitoring**: Whether or not to enable transaction monitoring.
    - **Supported Group Scheme**: A comma-separated list of URI schemes that the service treats as a group URI. This setting must be coordinated with the Group List Manager component configuration. The default value is `glmgroup`.

    For more information about these settings, refer to the topic *Group schema configuration*.
- **Services**: Default values used for Parlay X SMS over Parlay.
  - **SMS Service**: The following lists the configurable runtime properties:
    - **Purge Enabled**: True if cleanup activity occurs for expired delivery status data; false if cleanup activity does not occur. The default value is `false`.
    - **Purge Interval**: The time interval, in minutes, used to trigger the cleanup activity on Parlay X SMS over SMPP data. The default value is 10.
    - **Purge Block**: The number of records to retrieve each time there is a query to find expired data records. In effect, the cleanup operation is limited to

    cleaning up this much data during each purge interval. A value of 0 indicates that all records found should be removed. The default value is 1000.
  - **Service Enabled**: True if the service is enabled, false if disabled (when false, all requests are rejected). The default value is `true`.
  - **Network Resource Name**: The name of the network resource. Must match the name specified in **Network Resources** → **Network Resources Configuration** → *resource_name* for Parlay X SMS over SMPP. The default value is `PX21_SMS_SERVICE`.
  - **Default Requester for Orphan Messages**: The requester name to be included in orphan messages. In a notifySmsReception operation, an orphan message is defined as a message that is sent to a target where notification is not enabled, or a message in which the SMS destination does not fulfill the criteria specified in a previous startSmsNotification operation.
  - **Default Endpoint for Orphan Messages**: Endpoint to be notified when an orphan message is received.
  - **Default Correlator for Orphan Messages**: Correlator for the incoming orphan MO messages, used to correlate the messages while retrieving the received MO messages. The default value is an empty string, which indicates that any criteria will be matched. If you specify a non-empty string, then a given message will be retrieved only when a matching correlator is provided.
  - **Default Expiry Time for Orphan messages**: Amount of time, in minutes, to keep the orphan MO messages that have been received in the database. The default value is 1440.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

You can receive additional information on specific properties from the online help in the TWSS Administration Console.

### Policies

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X SMS over Parlay Web service in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X SMS over Parlay*.

## Administering Parlay X Call Handling over Parlay

After you deploy Parlay X Call Handling over Parlay, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default values used for Parlay X Call Handling over Parlay.
  - **Call Handling Web service**: The Parlay X Call Handling over Parlay Web service configuration settings in the Telecom Web Services Server Administration console are as follows:
    - **CallHandling Enabled**: Indicates if the CallHandling service is enabled.
    - **DetectOverLap**: Indicates whether the service detects the overlapping of regular address expressions.
    - **TextToSpeechAvailable**: Indicates whether the service accepts text as an input for processing with a Text-To-Speech engine.
    - **AudioContentAvailable**: Indicates whether the service accepts audio content for playing with an audio player.
    - **VoiceXMLAvailable**: Indicates whether the service accepts VoiceXML for processing with a VoiceXML browser.
    - **GroupSupport**: Indicates whether groups may be included with addresses.
    - **NestedGroupSupport**: Indicates whether nested groups are supported in group definitions.
    - **VoiceMailAvailable**: Indicates the availability of voice mail.
    - **Use Stored Procedure**: Indicates whether the stored procedure is enabled to store information in the database.
    - **Audio Format Supported**: Indicates the audio formats supported by the service, in a comma separated string format. For example; WAV, MP3, and AU.
    - **ParlayGatewayService**: The name string for the Parlay gateway service. The default value is: P_MULTI_PARTY_CALL_CONTROL.
    - **RetryCount**: When AutoRetry is set to *true*, this will indicate the number of retries you attempt when a Parlay command fails.
    - **AutoRetry**: Indicates whether or not the Parlay commands should be retried after a failed attempt.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X Call Handling over Parlay in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Call Handling over Parlay*.

## Administering Parlay X Call Notification over Parlay

After you deploy Parlay X Call Notification over Parlay, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

### TWSS Administration Console setting

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default runtime values for event notification.
  - **Call Notification Web Service**: This Web service handles calls initiated by a subscriber in the network.
    - **CallNotificationEnabled**: Whether the Call Notification Web service is enabled (true or false).
    - **NetworkResourceName**: The network resource name for the Web service implementation. This name should match the name defined in **Network Resources** → **Network Resources Configuration** → **Resource Name**.
  - **Call Notification Parlay**: Call Notification over Parlay service settings:
    - **ParlayServiceName**: The name string for the Parlay Gateway service. The default value is `P_MULTI_PARTY_CALL_CONTROL`.
    - **AutoRetry**: Whether Parlay commands should be retried on failure (true or false). The default value is `false`.
    - **RetryCount**: When AutoRetry is true, the number of retries to attempt when a Parlay command fails. The default value is `2`.
    - **RegisteredCallEventType**: An integer that designates the call event type registered for the Parlay Gateway for registered calls. The default value is `11`.
    - **RegisteredCalledNumberCallEventType**: An integer that designates the call event type for registered call-number calls. The default value is `2`.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

Parlay X Call Notification over Parlay does not use policies.

## Administering Parlay X Third Party Call over Parlay

After you deploy Parlay X Third Party Call over Parlay, you can use the TWSS Administration Console to change the deployment configuration properties for the Web service and related Service Platform components.

This component enables the application to send Web service requests to theTelecom Web Services Server, requesting the Parlay X Third Party Call over Parlay service.

### TWSS Administration Console settings

This Web service implementation adds the following configurable settings to the TWSS Administration Console:

- **Services**: Default values used for Parlay X Third Party Call over Parlay.
  - **Third Party Call Web Service**: Settings for configuring the Web service. The properties that can be configured include the following:
    - **StatusRetainTime**: The length of time in minutes to retain status after the termination of the call. This value must be a positive integer, or 0. The default value is 0.

    - **ChargingSupported**: Whether charging is supported for the MakeCall operation (`true`) or not (`false`). The default value is `false`.
    - **PurgeEnabled**: Whether cleanup activity occurs on expired delivery status data (`true`) or not (`false`). The default value is `false`.
    - **PurgeInterval**: The time interval in minutes that is used to trigger the cleanup activity on Third Party Call data. This value must be a positive integer. The default value is 10.
    - **PurgeDeleteBlock**: The maximum number of expired data records that can be removed during a single purge interval. The default value is 1000 records. (A value of 0 or less indicates that all records should be removed.)
  - **Third Party Call Parlay**: Parlay settings for the Third Party Call Web service. The properties that can be configured include the following:
    - **ParlayGatewayService**: The name string for the Parlay gateway service. The default value is `P_MULTI_PARTY_CALL_CONTROL`.
    - **AutoRetry**: Whether or Parlay commands should be retried on failure (`true`) or not (`false`). The default value is `false`..
    - **RetryCount**: When AutoRetry is true, the number of retries to attempt when a Parlay command fails. This value must be an integer between 0 and 65535. The default value is 2.
- **Common Components**: Click the name of a Service Platform component–for example **Fault Alarm Client** or **Usage Record Client**–to configure it to work with this Web service implementation. If you leave the endpoint URI empty for a component, then the component is disabled.

### Policies

The policies are configured through the Policy Management component. These policies will be assumed to be in the SOAP header and will be retrieved in the Web controller layer. For information about default policy configuration, refer to the following topic: *Default service policies for Parlay X Third Party Call over Parlay*.

## Maintaining the usage records database table

The usage records database table holds records that are generated by the Usage Record component Web service. Periodic maintenance of the table is recommended to optimize system performance. Maintenance consists of pruning the database table to remove files that are no longer needed.

The maintenance schedule depends on the database's transaction rate and the amount of space available in the database. In most instances it should be sufficient to prune the usage records database table once a week.

It is recommended that you perform the prune operation during off-peak hours.

**Note:** Pruning can be done on a live system.

The following sample Java application is provided to help you. It uses the default table name of `USAGERECORDS`. Before using the application, modify it as needed for your installation.

```
//IBM Sample Source Materials
//
//Sample source materials are supplied As-Is.
//No warranty is expressed or implied.
//
//(C)Copyright IBM Corp. 2000, 2006 , 2009
```

```
//
//The source code for this program supplied under the terms of the
//End User License Agreement (EULA) that accompanied this product.
//******************************************************************

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.NumberFormat;
import java.util.Date;
import java.util.Properties;

/**
* The TWSS Usage Records common component is a write-only common which produces usage records for
* each TWSS service used. These usage records are used for recording billing information.
* Once a usage record has been processed, the processed entry should be deleted from the database.
*
* This Sample Usage Record pruning class removes usages records from a database based on the records
* age. A SQL DELETE query is made of all the usage records older than the user provided input date a
* This query limits each delete request to 100 deletes, to provide user feedback, and also improve p
* issuing frequent database commits.
*
* Running the sample class CleanUpDbTable:
*
* Program assumptions:
*
*         - Run program on a TWSS server running environment.
*         - Run program inside a database command-line environment
*         - Suggested JVM version java 1.41
*
* Running the Sample:
*
* set CLASSPATH=.;%CLASSPATH%
* java  CleanUpDbTable  DB2 TWSSDB62 localhost 50000 userid password 2007-09-14 11:00:00 TableName T
*
* or
*
* java   -cp .;%CLASSPATH% CleanUpDbTable.class   DB2 TWSSDB62 localhost 50000 userid password 2007-
*
*
* Typical runtime problems:
*
*         Error:  ClassNotFoundException
*         Action: Ensure WAS Server and database is running or Program can't find JDBC driver in cla
*
*         Error:  Incorrect Userid or Password
*         Action: Ensure correct information , recheck command line positions.
*
*         Error:  SQLException errors
*         Action: Check proper date or time command line input.
*

 * @author Administrator
 */
public class CleanUpDbTable {


 /**
  * Error messages
  */
 static final String ILLEGAL_ACCESS = "IllegalAccessException during driver resolution: " ;
 static final String INSTANTIATION_EXCEPTION = "InstantiationException during driver resolution: ";
 static final String CLASS_NOT_FOUND = "ClassNotFoundException during driver resolution:  ";
 static final String SQL_DRIVER = "SQLException during driver getConnection:  ";
static final String SQL_COUNT = "SQLException during UseageRecord count query: ";
static final String SQL_EXECUTE = "SQLException during query executeUpdate: ";
static final String SQL_DELETE =  "SQLException during query delete processing: " ;
static final String SQL_CLOSE = "Error closing connection: " ;

/**
 * Informational messages
 */
static final String USAGE1 = "Usage: database-alias server-hostname portNumber userId password da
 "Example parameters: [DB2|ORACLE] WPS_AL dbserverhostname 50000 username <password> 2007-09-01 1
static final String MSG1  = "Starting query count of usage records to delete...";
static final String MSG2  = "SQL Query = \n";
static final String MSG3  = "Usage Records = " ;
static final String MSG4  = "Committed delete UsageRecords row(s): ";
static final String MSG5  = "Current elapsed time is: ";
static final String MSG6  = "Averaging delete time: " ;
static final String MSG7  = "Total UsageRecord cleanup time:  ";
static final String MSG8  = "UsageRecord total delete count query time: " ;
static final String MSG9  = "UsageRecord count query matching row(s) count: ";
static final String MSG10 = "Starting Delete usage query...";
static final String MSG11 = "Preparing jdbc URL: ";
static final String MSG12 = "Preparing jdbc driver class: ";
static final String MSG13 = "No UsageRecords found matching query criteria. ";

/**
 * Remember database type
 */
static boolean isDB2 = false;
static boolean isOracle = false;


/**
 * URL property file inputs
 */
static final String USERID = "user";
static final String PW = "password";

/**
 * CommandLine parameter inputs
 */
static final int _DB = 0 ;
static final int _ALIAS = 1 ;
static final int _SERVER = 2 ;
static final int _PORTNUM = 3 ;
static final int _USERID = 4 ;
static final int _PW = 5 ;
static final int _TIME = 6 ;
static final int _DATE = 7 ;
  static final int _TABLE = 8 ;
  static final int _TIMEFIELD = 9 ;
  static final int _PRIMARYKEY = 10 ;
static final int MAX_PARM = 11;

/**
 * Java Main Application entry point
 *
 * @param args
 */
public static void main(String[] args) {

 CleanUpDbTable example = new CleanUpDbTable();

 // check for parameters
 if (args.length > MAX_PARM || args.length < MAX_PARM) {
  example.myexit (-1 , USAGE1 );
 }
```

```
example.processCmd(args);

String alias = args[_ALIAS] ;
String server = args[_SERVER];
int portNumber = Integer.valueOf(args[_PORTNUM]).intValue();
String userId = args[_USERID];
String password = args[_PW];
    String tableName = args[_TABLE];
    String timeFieldName = args[_TIMEFIELD];
    String primaryKeyName = args[_PRIMARYKEY];

// for collecting some simple statistics
Date startTime = new Date();
int count = 0;
int records = 0;
Connection connection = null;
String url = null;
String driver = null;
if (isDB2) {
 url = "jdbc:db2://" + server + ":" + portNumber + "/" + alias;
 driver = "com.ibm.db2.jcc.DB2Driver";
} else {
 url = "jdbc:oracle:thin:@" + server + ":" + portNumber + ":" + alias;
 driver = "oracle.jdbc.driver.OracleDriver";
}

// MSG11 = "Preparing jdbc URL: ";
// MSG12 = "Preparing jdbc driver class: ";
example.tracemsg (MSG11 + url);
example.tracemsg (MSG12 + driver);


try {
 Class.forName(driver).newInstance();
} catch (IllegalAccessException e) {
         e.printStackTrace();
 example.myexit(-2,ILLEGAL_ACCESS + e);
} catch (InstantiationException e) {
         e.printStackTrace();
 example.myexit(-2,INSTANTIATION_EXCEPTION + e);
} catch (ClassNotFoundException e) {
         e.printStackTrace();
 example.myexit(-2,CLASS_NOT_FOUND + e);
}

Properties props = new Properties();
props.setProperty(USERID, userId);
props.setProperty(PW, password);

try {
 connection = DriverManager.getConnection(url, props);
 connection.setAutoCommit(false);
} catch (SQLException e) {
         e.printStackTrace();
 example.myexit(-3,SQL_DRIVER + e);
}

if (connection != null) {

 try {
  // msg1: "Starting count of usage records to delete...";
  example.tracemsg (MSG1);
  //-----------------------------------------------------
  // Query the DB for a total all records older
  // than the time/date provided
  //----------------------------------------------------
  Statement statement = connection.createStatement();
 String query = "SELECT  COUNT( " + timeFieldName + " ) AS TOTAL  FROM " + tableName + " WHERE
  + args[_TIME] + " "
  + args[_DATE].trim() + "'))";
 // msg2:"Using query: ";
 example.tracemsg (MSG2+ query);


 //--------------------------------
 // Perform Usage Record Count
 //--------------------------------
 ResultSet rs = statement.executeQuery(query);
 rs.next();
 records = new Integer(rs.getInt("TOTAL")).intValue()  ;

 // records = Integer.parseInt(recordCountString);
 // msg3:"Usage Records = " + records
 example.tracemsg (MSG3+ records);
 rs.close();
 statement.close();

} catch (SQLException e) {
          e.printStackTrace();
 example.myexit(-4,SQL_COUNT + e);
}

Date finshTime = new Date();
// MSG8 = "UsageRecord total delete count query time: " ;
// MSG9 = "UsageRecord count query matching row(s) count:";
example.tracemsg (MSG8+  example.timeInSeconds(finshTime.getTime() - startTime.getTime()));
example.tracemsg (MSG9+records)  ;


// Check if any delete matching criteria
if (records != 0) {

 // Prepare to delete and commit 100 Usage records per loop
 count = 0;
 startTime = new Date();

 //msg10 = "Starting Delete usage query...";
 example.tracemsg (MSG10)  ;

    String query = "DELETE FROM " + tableName + " WHERE " + primaryKeyName + " in (SELECT " + p
     + args[_TIME] + " "
     + args[_DATE].trim() + "') ORDER BY " + primaryKeyName + " FETCH FIRST 100 ROWS ONLY)";

 example.tracemsg (MSG2+ query);

 try {
  PreparedStatement statement = connection.prepareStatement(query);
  Date runningTime = new Date();

  int delcnt = 0 ;
  while (records > 0) {

   try {

    delcnt = 0;
    //--------------------------------
    // Perform Usage Record Delete
    //--------------------------------
    delcnt= statement.executeUpdate();

   } catch (SQLException e) {
             e.printStackTrace();
    example.myexit(-4,SQL_EXECUTE + e);
```

```
     }
     count+= delcnt;
     records -= delcnt;

     //---------------------------------------------
     // To prevent accessive locks commit often and
     // improves preformance
     //---------------------------------------------
     connection.commit();

     finshTime = new Date();
     long intermediateTime = finshTime.getTime() - runningTime.getTime();


     // msg4: "Committed delete UsageRecords row(s): " +delcnt
     // msg5: "Current elapsed time is: " + example.timeInSeconds((finshTime.getTime() - startTime.g
     // msg6: "Averaging delete time in seconds: " + example.timeInSeconds(intermediateTime / 100))
     example.tracemsg (MSG4+ delcnt );
     example.tracemsg (MSG5+  example.timeInSeconds( (finshTime.getTime() - startTime.getTime()) )
     example.tracemsg (MSG6+  example.timeInSeconds(intermediateTime / 100));

     runningTime = new Date();
    }

    statement.close();

   } catch (SQLException e) {
             e.printStackTrace();
    example.myexit(-4,SQL_DELETE + e);
   }

   finshTime = new Date();
   // msg7  "Total UsageRecord cleanup time:  ";
   example.tracemsg (MSG7+ example.timeInSeconds(finshTime.getTime() - startTime.getTime())) ;

  } else { // if count
   // MSG13 = "No UsageRecords found matching query criteria. ";
   example.tracemsg (MSG13) ;

  }
 } // if connection
 if (connection != null) {
  // Disconnecting ...
  try {
   connection.commit();
   connection.close();
  } catch (SQLException e) {
            e.printStackTrace();
   example.myexit(-4,SQL_CLOSE + e);

  }
 }

 example.myexit(0,"");

}

/** A number formatter*/
private NumberFormat nf = NumberFormat.getInstance();

/**
 * Convert time in milliseconds to a String in seconds
 * @param time
 * @return
 */
private  String timeInSeconds(float time) {
 return nf.format(time/1000F) + " seconds";
 }


 /**
  * processCmd:  Check input parms.
  * @param args  Program input parms.
  */

 private void processCmd (String[] args ){

  // Did the user request help?
  if ( (args[_DB].indexOf("?")> -1) || (args[_DB].trim().toLowerCase().indexOf("help")> -1)  ) {
   myexit (-1 , USAGE1 );

   // Check for DB2
  }  else if (  (args[_DB].trim().toUpperCase().indexOf("DB2") > -1) ) {
   isDB2 = true;
   // Check for Oracle
  }  else if (  (args[_DB].trim().toUpperCase().indexOf("ORACLE") > -1) ) {

  } else {
   myexit (-1 , "" );

  }

 }



 /**
  * tracemsg: Output messages
  * @param message display message
  */

 private void tracemsg (String message ){

  System.out.println(String.valueOf(message));
 }

 /**
  * Exit: System exit
  * @param retc    return error code
  * @param errmsg  display message
  */

 private void myexit(int retc , String errmsg  ){

  tracemsg(String.valueOf(errmsg));
  System.exit(retc);

 }

}
```

**Note:** You may also find that system performance is improved by performing similar pruning operations on your other TWSS databases tables, for example NETWORKSTATISTICS.

## Using IBM Tivoli License Manager

The IBM Tivoli License Manager (ITLM) product is used to detect where IBM products are both installed and running. ITLM is installed with each of the IBM WebSphere software for Telecom products.

## Compatibility

This release of IBM WebSphere Telecom Web Services Server (TWSS) runs with ITLM server version 2.2.

**Note:** The ITLM agent version 2.2 might not be compatible with all versions of the operating systems that are supported by TWSS. Review the ITLM documentation carefully to determine which operating systems, maintenance levels, and Linux kernel versions are supported.

## Installation

During the installation of TWSS, the inventory signatures for ITLM are installed in the `*was_profile_root*/installedApps` directory.

## Inventory Signature

The ITLM inventory signature file uniquely identifies the product and is installed with the TWSS EAR file, which is treated as an application by WebSphere Application Server. The inventory signature is located within the EAR file in the `TIVREADY` subdirectory, and its name is `IMSTWS0700.SYS2`.

## License file

Each TWSS subcomponent comes with a license file that is not associated with ITLM enablement, but is still important for licensing purposes. Each of these text files specifies the customer's entitlement of installation and use of a specific TWSS subcomponent.

## Usage Signature

The ITLM usage signature, generated by the Software Catalogue Signature team, is used to identify each component as a J2EE product.

# Chapter 6. Integrating and Developing with WebSphere Telecom Web Services Server

Documentation is provided to help you integrate with and take advantage of services offered by the Telecom Web Services Server (TWSS) platform.

Call flows and interface information are provided for the Web service implementations and the TWSS subcomponents. The following topics contain more information.

Javadoc is included for the TWSS Administration Console to enable integration into your network. To access the Javadoc, use the link at the end of this page.

To access WSDL documentation for the Service Platform components, use the link at the end of this page.

## Integrating with the Notification Management component Web service

The Notification Management component Web service provides notification tracking and administrative functionality.

### Notification Management implementation

You can program an administrative user interface to the Notification Management component Web service. Each Web service operation is defined by WSDL.

#### Call flows

All Web service requests and responses pass through and are inspected by the Access Gateway. The Access Gateway acts as an intermediary for all communication between the client and service implementation endpoints. The incoming requests to the Service Platform and the outgoing requests to the notification will be processed.

The following steps show an example of the call flow for the incoming Service Platform request:

1. An incoming service request is initiated.
2. The Access Gateway uses policy information from the Service Policy Manager component to select the service implementation endpoint.
3. The service implementation endpoint is selected, then invoked.
4. The service implementation calls the Notification Manager, to inform it of notification registrations that have been started or ended.

The following steps show an example of the call flow for the outbound Notification request:

1. The service implementation issues an outbound request.
2. The request is provisioned through the Notification Management component Web service.
3. As the notification event occurs, the core network notifies the Web service implementation. The Web service implementation calls PX Notification component Web service, which delivers an event through the Access Gateway

to the application. The PX Notification also contacts the Notification Manager statistics interface to provide information about the number of notification events that have been delivered.

### Interfaces

Following is a list of interfaces defined for the Notification Management component Web service.

**NotificationRegistration**
Web service implementations use this interface to register with and remove notifications from the Notification Management component Web service.

**NotificationAdministrationSupport**
Services that choose to allow administrative termination of a notification must implement this interface. When an administrator chooses to terminate a notification through the NotificationAdministration interface, the Notification Management component Web service uses the NotificationAdministrationSupport interface to interact with the Web service implementation. It then terminates the notification.

**NotificationStatisticsPublishing**
This interface allows Web service implementations or other Service Platform components to publish success and failure statistics on notification deliveries with the Notification Management component Web service.

For all Parlay X-based Web service implementations, the statistics are published through the PX Notification component Web service. The Notification Management component Web service makes statistics available to administrative users through the NotificationAdministration interface.

**NotificationAdministration**
Administrative tools or users use this interface to access registered notifications, gather published statistics about notification delivery, and terminate notifications. This interface can be used for both interactive investigation and bulk operational work.

## Removing notifications

Notifications can be removed by issuing a RemoveNotification request from the Notification Management component Web service common component.

### Before you begin

The following operations issue a RemoveNotification request:
- CallNotificationManager.stopNotification
- CallDirectionManager.stopNotification
- NotificationAdministrationSupport.terminateNotification

Complete the following steps to remove a notification:
1. Stop the notification method implementation in the Web service. This method will remove the notification related data that is entered in the database tables related to the Notification Management component Web service.

Complete the following steps to terminate a notification:
1. The administrator invokes this method. The method will not remove the notification related data entered in the database tables related to the

Notification Management component Web service. The administrator has to delete the notification related data explicitly.

## Interacting with SIP network elements

SIP network elements use P-Charging-Vector headers to collect charging information for communications that flow across one or more networks. Your Telecom Web Services Server applications can generate information about the network elements on which they run and collect information about other network elements with which they communicate.

### Originating (initiating) the SIP request

Applications running on network elements that originate SIP traffic, such as creating new and outbound phone calls, can add P-Charging-Vector headers to their outbound SIP requests and responses.

#### Before you begin

Your application should accept and use the correct IOI (Inter Operator Identifier) as a configuration parameter.

#### About this task

The following steps describe the process by which an application adds P-Charging-Vector headers to outbound SIP requests and responses:

1. The application receives the initial incoming **SipServletRequest**.
2. The application invokes the **PChargingVector.piggybackOnto(...)** method, using the incoming SipServletRequest.

   **Note:** You must supply an Inter Operator Identifier (IOI) to the method. The IOI will be used as the terminating IOI.
3. The application generates a **SipServletResponse** to the SipServletRequest.
4. The application invokes the **PChargingVector.piggybackOnto(...)** method, on the newly generated SipServletResponse.
5. The application sends the **SipServletResponse**.
6. The application invokes the **PChargingVector.piggybackOnto(...)** on all further SipServletRequests or SipServletResponses received or generated by the application.

   **Note:** The application must use the Java standard locking mechanisms to ensure that the **PChargingVector.piggybackOnto(...)** is not invoked on two different messages in the same SIP dialog at the same time. Locking on the **SipServletMessages** SipSession prior to calling **PChargingVector.piggybackOnto(...)**, is sufficient.

### Terminating (answering) the SIP request

Applications running on network elements that terminate SIP traffic, such as those that answer incoming phone calls, can collect and add to the information present in the P-Charging-Vector headers.

### Before you begin

Your application should accept and use the correct Inter Operator Identifier (IOI), as a configuration parameter.

### About this task

SIP border and privacy proxy elements can create call accounting records using the P-Charging-Vector information provided by the originating and terminating elements. The headers typically include the following data:

- The name of the host that created the header
- The IMS Charging Identifier (ICID)
- The originating network IOI
- A terminating network IOI

**Note:** The charge header support vector is a utility class for handling the SIP messaging for charging interactions.

The following steps describe the process by which your application can handle P-Charging-Vector headers in inbound SIP requests and responses:

1. The application receives an initial incoming **SipServletRequest**.
2. The application invokes the**PChargingVector.piggybackOnto(...)** method using the incoming SipServletRequest.

   **Note:** You must provide an IOI to the method. This IOI will be used as the terminating IOI
3. The application generates a **SipServletResponse** to the SipServletRequest.
4. The application invokes the **PChargingVector.piggybackOnto(...)** method on the newly generated SipServletResponse.
5. The application sends the **SipServletResponse**.
6. The application invokes the **PChargingVector.piggybackOnto(...)** on all further SipServletRequests, or SipServletResponses received or generated by the application.

   **Note:** The application must use the Java thread synchronization mechanisms to ensure that the **PChargingVector.piggybackOnto(...)** is not invoked on two different messages in the same SIP dialog at the same time. Locking on the **SipServletMessages** SipSession prior to calling **PChargingVector.piggybackOnto(...)**, is sufficient.

## Custom JCA adapter

TWSS 7.1 allows the ability to provide a JCA adapter for the purposes of achieving pluggable component architecture for connection management.

The following are features and benefits of the ability to customize your JCA adapter:

- Make the Web service implementation layer reusable with any JCA adapter (based on native protocols like SMPP, CMPP, and, UCP) that adheres to JCA 1.5 specification. The Web service implementation layer targets both SMS SMPP and WAPPush SMPP Web Services.

- Make the SMPP JCA adapter pluggable to any Web service implementation layer that is developed based on the Parlay X v2.1 specification for the SMS Web service.
- Expose public APIs for developing custom JCA adapters for native protocols like CMPP and UCP.

## Developing a custom JCA adapter

To develop a JCA adapter for other native protocols, create a new class and then associate objects with the new class.

1. Create a <ProtocolName> MessageTranslator class extending the com.ibm.mds.adpt.sms.smpp.util.SmsMessageTranslator class.
   - This class parses the JCA records corresponding to Parlay X operations.
   - It constructs the protocol specific PDUs
2. From the Web service implementation (SI) layer, define the BaseInteractionSpec object to be used for propagating the function name attribute to the adapter.
3. From the SI layer, define the BaseConnectionSpec object to be used for propagating the following attributes corresponding to the Network Element to the adapter:
   - Host name
   - Port number
   - System ID // Username
   - Password corresponding to the system ID
   - Protocol ID // Protocol version
   - Bind type (transmitter, reciver, or transeiver)
   - Message type (synchronous, statusless, and asynchronous)
   - Assignment ID

   These attributes are initialized to null values, so that if a particular JCA implementation does not require any of the fields, you can create a BaseConnectionSpec object using its default constructor and initialize the values or retrieve the values only for the required attributes using the setter/getter methods.
4. Create an interaction object implementing the javax.resource.cci.Interaction interface, as shown in the following sample code:

```
public class <ProtocolName>lInteration implments javax.resource.cci.Interaction {

public void close() throws javax.resource..ResourceException {
    // Close the connections established with the Network Entity
}

public Connection getConnection() {
// Retrieves the object corresponding to the connection
//established with the SMSC
}

public boolean execute(InteractionSpec arg0, Record request, Record response)throws ResourceEx

      // Type cast the InteractionSpec to BaseInteractionSpec type.
     BaseInteractionSpec csSepc = (BaseInteractionSpec) ispec;

    // Extract the function Name from the InteractionSpec
     String functionName = csSepc.getFunctionName();

   // Check the function name for the values
// SendMessage
// BindToNetworkElement
```

```
  // StartMOMessages
// MonitorHeartBeat
// QueryMessageStatus

if(functionName.equals("BindToNetworkElement ") || functionName.equals("StartMOMessages ")){

// Parse the Record corresponding to the function
BindToNetworkElement
bindRecordFrmReq = (BindToNetworkElement)record;

// Construct the JCA Records corresponding to the native protocol

// Establish a connection with the Network Element

// Construct the JCA Record corresponding to Parlay X adding the conection status

// Send the response back to SI Layer

}
if(functionName.equals("SendMessage"){

// Parse the Record corresponding to Parlay X
SendSMSRecord smsRecord = (SendSMSRecord)record;

// Construct the JCA Records corresponding to the native protocol using the MessageTranslator obj

// Send the message to the Network Element

// Send the response back to SI Layer

}

if(functionName.equals("MonitorHeartBeat "){

// Parse the Record corresponding to Parlay X
MonitorHeartBeat heartBeatRecord = (MonitorHeartBeat)record;

// Construct the JCA Records corresponding to the native protocol using the
// MessageTranslator object

// Send the message to the Network Element

// Send the response with the connection status to SI Layer

}
if(functionName.equals("QueryMessageStatus"){

// Parse the Record corresponding to Parlay X
QueryMessageStatusRequest queryStatusRecord = (QueryMessageStatusRequest)record;
// Construct the JCA Records corresponding to the native protocol using the
// MessageTranslator object

// Send the message to the Network Element

// Send the response with the message status to SI Layer

}

}
```

5. To handle Mobile Origination (MO) messages, perform the following steps:
   a. The MO messages (either of type confirm delivery for a sendSMS mesage or plain MO messages related to the notifications already started) need to be read on the socket connections already opened during sendSMS or startNotification.
   b. Parse the protocol specific records, for example deliverSM in SMPP.

c. Convert them to the JCA records, of type "MOMessageRequest."
d. Create a <ProtoclName>Message object, which implements the javax.jms.Message interface.
e. Add the MOMessageRequest (of type JCA record ) as an object property to the Message object.
f. Create a MessageLister object (which will be part of the SI layer) that is of type javax.jms.MessageListener. Typically this MessageListener object is a message driven bean in J2EE terms.
g. Add the Message object as a request parameter to the onMessage method in the MessageListener object, and call the onMessage method.
h. To address the scalability requirement, it is required to create a proxy of the endpoint for every MO message received. Handle it and release the proxy.

## JCA adapter functions

The isolated JCA adapter functions must be invoked through execute methods in the implemented classes of the "javax.resource.cci.Interaction" interface.

**SendMessage**

- This function is used to send the mobile terminal messages to mobile terminals.
- It is an interface for the operations SendSMS, SendSMSLogo, SendSMSRingTone, and SendWAPPushSI in the Web service implementation layer.
- It is an interface for the PDU's SUBMIT_SM and SUBMIT_MULTI in the SMPP JCA adapter.

**QueryMessageStatus**

- This function is used to send query message status requests to SMSC.
- It is an interface for the operation GetSmsDeliveryStatus in the Web service implementation layer.
- It is an interface for the PDU QUERY_SM in the SMPP JCA adapter.
- In TWSS 7.1, SMS Web service implementation does not provide an implementation for this function. This function is exposed as a place holder for future references for custom Web service implementation for those who want to interact with network element for retrieving the latest status of the sent message.

**StartMOMessages**

- This function establishes a connection with the network element.
- The connection is based on the TCP/IP protocol.
- It is an interface for the operation StartSmsNotification in the Web service implementation layer.
- It is an interface to the PDU's BIND_RECEIVER and BIND_TRANSCEIVER in the SMPP JCA adapter.

**BindToNE**

- This function establishes a connection with the network element.
- The connection is based on the TCP/IP protocol.
- The Web service implementation layer does not have a direct interface to this function in the form of WSDL.

- It is an interface for the PDU's BIND_TRANSMITTER, BIND_RECEIVER and BIND_TRANSCEIVER in the SMPP JCA adapter.
- In the SMS Web service implementation and SMPP JCA implementation, the binding is established as part of sending a short message to NE or while starting SMS notifications.
- The SMS Web service implementation layer does not make an explicit call to this function in the adapter. At present, from the Web service implementation layer, the connection establishment is done as a prerequisite to the sendSMS operation.
- This function is exposed as a place holder to address design approaches where an explicit connection is employed.

**UnbindNE**

- This function removes or closes the connection established with the network element.
- The connection is based on the TCP/IP protocol.
- The Web service implementation layer does not have a direct interface to this function in the form of WSDL. Though stopSmsNotification exists in the Parlay X SMS Web service specification, the UnbindNE is not called by the SMS Web service implementation for performing a stopSmsNotification operation because the established connection might be used for notifications and for sending mobile terminal messages.
- It is an interface to the PDU's UNBIND in the SMPP JCA adapter.
- In the SMS SI and SMPP JCA implementation, the unbind happens as part of cleaning the connections during scenarios such as the system going down or stopping the cluster.

**MonitorHeartBeat**

- This function monitors the status of the connection between the Web service implementation and the network element.
- The connection is based on the TCP/IP protocol.
- The Web service implementation layer does not have a direct interface to this function in the form of WSDL.
- In the SMS SI, there is an MBean attribute to initiate the heart beat monitor feature. The frequency for sending signals to monitor the connection is configurable.
- It is an interface to the PDU's ENQUIRE_LINK in the SMPP JCA adapter.
- At present in the Web service implementation layer, the heart beat monitor is used as a means to validate the connection with a network element before opening a new connection with a network element.

## JCA adapter records

These records are Java Beans and implement the "javax.resource.cci.Record" interface in JCA v1.5 specification. These records behave as information carriers between the Web service implementation and the adapter for communicating the attributes corresponding to respective adapter exposed functions.

**SendMessageRequest**

- This record is used while sending the mobile terminal messages to mobile terminals.

- It is used while performing the Parlay X operations SendSms, SendSmsLogo, and SendSmsRingTone, as well as the WAP10 operation SendWAPPushSI.
- The attributes for this record are:
  - addresses
  - sourceAddress
  - senderName
  - textMessage
  - binaryMessage
  - segmentNumber
  - numberOfSegments
  - functionID
  - charging
  - chargingCodeFromServiceType
  - serviceType
  - billingCode
  - serviceTypeUsage
  - chargeForFirstSegmentOnly
  - sourceTon
  - sourceNpi
  - destinationTon
  - destinationNpi
  - messageMode
  - messageType
  - nwFeature ( UDHI or not )
  - protocolID
  - priorityFlag
  - replaceIfPresentFlag
  - messageIDFormat
  - encoding
  - operationName
  - isBinary

**SendMessageResponse**

- This record is used by the adapter when it sends the response messages to the Web service implementation layer corresponding to the mobile terminal messages sent to the mobile terminals.
- It is used while performing the Parlay X operations SendSms, SendSmsLogo, and SendSmsRingTone.
- The attribute for this record is:
  - messageID

**BindToNERequest**

- This record is used while the Web service implementation layer sends a request to the JCA for establishing a connection with the network element.
- There is no corresponding Parlay X Web service operation for using this record.

- The attributes for this record are:
  - bindType
  - systemID
  - password
  - systemType
  - functionID
  - hostname
  - portNumber

**BindToNEResponse**

- This record is used while the JCA layer sends the response to the Web service implementation upon establishing the connection with the network element.
- There is no corresponding Parlay X Web service operation for using this record.
- The attribute for this record is:
  - bindSucceeded

**UnbindNERequest**

- This record is used while the Web service implementation layer sends a request to the JCA for closing already established connections with the network element.
- This record is used when calling the UnbindNE function in the adapter.
- The attribute for this record is:
  - functionID

**UnbindNEResponse**

- This record is used while the JCA layer sends the response to Web service implementation when closing the existing connections with the network element.
- The attribute for this record is:
  - unbindSucceeded

**MonitorHeartBeat**

- This record is used while sending the request message corresponding to the heart beat monitoring operation.
- The attribute for this record is:
  - functionID

**MOMessage**

- This record is used while the JCA layer forwards the mobile origination messages from the network element to the Web service implementation layer.
- The attribute for this record is:
  - functionID
  - sourceAddress
  - sourceAddressTon
  - sourceAddressNpi
  - destinationAddress
  - destinationAddressTon
  - destinationAddressNpi

  - messageType (Delivery_Receipt, Read_Ack, Normal_MO)
  - moMessage
  - messageState (DELIVERED, EXPIRED, DELETED, UNDELIVERED, ACCEPTED, UNKNOWN, REJECTED)
  - receievedMessageID
  - networkErrorCode
  - serviceType
  - protocolID
  - priorityFlag
  - registeredDelivery
  - encodingScheme
  - messageLength

**QueryMessageStatusRequest**

- This record is used while the Web service implementation layer queries the network element for the status of the already sent message.
- The attributes for this record are:
  - functionID
  - sourceAddress
  - messageID
  - sourceAddressTon
  - sourceAddressNpi

**QueryMessageStatusResponse**

- This record is used while the network element sends the status of the already sent message to the Web service implementation.
- The attributes for this record are:
  - messageID
  - messageState
  - finalState
  - networkErrorCode

**ErrorMessage**

- This record is used while the Web service implementation layer sends an error message as a response to the network element.
- The attributes for this record are:
  - functionID
  - errorMessage

## Sample code to allow the Web service implementation to interact with the JCA adapter

Use this sample code to get the Web service implementation layer to interact with the JCA adapter.

The following example code allows SMS/SMPP to interact with the JCA adapter. To use this for WAP Push, change the object names accordingly.

```
// Create a BaseConnectionSpec object by providing the NE related details
BaseConnectionSpec spec = createConnectionSpec(/*<An object which carries data related to NE conne

// Create a BaseInteractionSpec object & set the function name
```

```
BaseInteractionSpec ispec = new  BaseInteractionSpec();
ispec.setFunctionName("SendMessage");

// Retrieve the InitialContext
InitialContext ic = new InitialContext();

// Create a Connection Factory
connectionFactory = (ConnectionFactory)ic
    .lookup("java:comp/env/eis/SMPP_CF");

// Retrieve the Connection object from WAS & type casting it to
// SMPPConnection object (An object which implements the JCA Connection // interface)
Connection cx = (SMPPConnection)connectionFactory.getConnection(spec);

// Create an Interaction object & type cast it
Interaction ix = (SMPPInteraction) cx.createInteraction() ;

// Create the JCA Record object corresponding to JCA Function Name

// Call the execute method in the Interaction object
ix.execute(ispec, /*<created JCA Record&gt;*/);
```

# Open network enablers API

The open network enablers (ONE) API is a single API that can be used across multiple operators.

The Access Gateway currently supports only Web service invocations to access the telecom services provided by the service platform. The purpose of the ONE API is to extend the channel support for Telecom Web Services Server Access Gateway to include HTTP.

OneAPI, being a Parlay X derivative, is easily mapped to the existing Parlay X 2.1 Access Gateway flows.

The following principles apply to the ONE API:

- Services should be defined as entities or resources and URLs defined accordingly, for example using nouns not verbs. Messages, subscribers, and calls become resources.
- Use HTTP verbs, such as POST, GET, PUT, and DELETE, for all interfaces.

Dojo widgets and forms are provided for each supported .operation, and are provided in the WebSphere Telecom Web Services Server v7.1 toolkit.

## Message flows

Both mobile terminated message flows as well as mobile originated message flow may use the ONE API.

### Mobile terminated message flows

A Web application may access common telecom functionalities using ONE API. For example, a ONE API call can be made from a Dojo widget. The address and message fields are sent as part of the HTTPHeader. Two new export components have been introduced for each of the supported interfaces, one for the XML content-type and another for JSON.

A custom handler configured in the export component converts the message from native HTTP form data into an SDO for handling a request. Likewise, it is converted from an SDO into a minimal XML string (as specified by the One API) for handling the response. The SCA HTTP binding was introduced in WebSphere Integration Developer v6.1.

A function selector determines what operation to call on an export interface. When the HTTP export receives a request message, it uses the function selector to determine the method name that it needs to invoke on the interface based on the HTTP request information.

The following is an example of a typical request JSON message for sendSms:

```
{
"address": ["9845054321", "9632112333"],
"senderName": "JDoe",
"charging": {
             "currency": "INR",
             "amount": "0.50",
             "code": "SMS- national"
      },
 "message": "how are you?",
 "receiptRequest": {}
};

A typical response JSON message for sendSms would look as shown below
{
"messageId": "123bca"
};
```

### Mobile originated message flows

Notification type operations, such as notifySmsReception, are implemented on the application side. In this case, the notification message passes through the Iimport component SMS_OneAPIImport. An SCA HTTP import binding enables SCA applications to directly invoke external services over HTTP. This import component is responsible for transforming the data from SDO to the native HTTP formatted message at the application side, implemented as per ONE API specifications.

Since there are now two imports implementing the same interface, a routing decision must be made in case the message is intended for the ONE API import. So in the notification flows (operations), a small change must be made.

When a new partner is added to the WebSphere Integration Developer's Operation Connections pane, it adds a new callout. A filter has been added that makes the decision as to whether this message is intended for the ONE API endpoint, JSON endpoint, or the standard Web service endpoint.

## Interfaces

Interfaces and operations for the ONE API.

These interfaces and operations are standard-based specifications. Consult the standards documentation for details.

In the event that mandatory fields expected by the Parlay X 2.1 specifications are not present in the ONE API specification, the default value will be populated for that field.

### Location interface

The operation available for the Location interface through the ONE API.

### getLocation via HTTP-GET

Based on the Parlay X 2.1 operation of the same name, this operation allows you to ask the network for the location of a terminal (or terminals).

#### Sample request

```
GET /<path>/getLocation?address=tel:+447990123456&
address=tel:+447990121212&accuracy=100 HTTP/1.1
Host: www.example.com
```

**Note:** To keep the method signature simple, acceptableAccuracy, maximumAge, responseTime, and tolerance are not included in the REST version of getLocation.

#### Sample response

```
HTTP/1.1 200 OK
-------------
<GetLocationResponse version="1.0">
<location address="tel:+447990123456" timestamp="Tue, 15 Nov 1994 08:12:31 GMT"
      longitude="-5.932617" latitude="54.601505" altitude="10.0" accuracy="100"/>
<location address="tel:+447990121212"  timestamp="Tue, 15 Nov 1994 08:12:31 GMT"
      longitude="-6.03581" latitude="54.782351" altitude="10.0" accuracy="100"/>
</GetLocationResponse>
```

## Payment interface

Operations available for the Payment interface through the ONE API. All of these operations are based on the Parlay X 2.1 standard.

### chargeAmount via POST

Used for charging an amount to an end user's account.

#### Sample request

```
POST/path/chargeAmount?version=1.0&endUserId=tel:+447990123456&currency=GBP&amount=1&referenceCode=A
Host: www.example.com
```

#### Sample response

```
HTTP Status: 204 No Content.
```

**Note:** Either currency and amount or code should be specified. If both are specified, the exception ″SVC0007 – Invalid charging information″ is thrown.

### refundAmount via POST

Used for refunding an amount to an end user' account

#### Sample request

```
POST /path/refundAmount?version=1.0&endUserId=tel:+447990123456&
currency=GBP&amount=1&referenceCode=ABC&
description= Some billing information
HTTP/1.1 Host: www.example.com
```

#### Sample response

```
HTTP Status: 204 No Content.
```

### reserveAmount via POST

Used for reserving a charge for an end user’s account.

#### Sample request

```
POST /path/reserveAmount?version=1.0&endUserId=tel:+447990123456&
currency=GBP&amount=10&
description= some billing information
HTTP/1.1 Host: www.example.com
```

#### Sample response

```
HTTP Status: 201 Created
<ReserveResponse version="1.0" reservationId="1234"/>
```

### chargeReservation via POST

Used for Charging to a previously made reservation

#### Sample request

```
POST /<path>/chargeReservation/1234/charge?version=1.0&
currency=GBP&amount=6&referenceCode=ABC&
description=Some billing information
HTTP/1.1 Host: www.example.com
```

#### Sample response

```
HTTP Status: 204 No Content.
```

### releaseReservation via DELETE

Used for releasing funds left in a previously made reservation to the account from which the reservation was made.

#### Sample request

```
DELETE /<path>/releaseReservation/1234/release?version=1.0
HTTP/1.1 Host: www.example.com
```

#### Sample response

```
HTTP Status: 204 No Content.
```

## SMS interface

Operations available for the SMS interface through the ONE API. All of these operations are based on the Parlay X 2.1 standard.

### sendMessage via POST

Provides a subset of that standard’s functionality. This interface is for sending an SMS to a terminal, for example from a Web page or desktop application. This lets you send an SMS to one or more terminals, for example mobile phones or SMS-enabled laptops.

#### Sample request

```
POST  /<path>/senSms?version=1.0& address=tel:+447990123456& address=tel:+447990121212& message=He
correlator=123456&
notifyURL=http://myapp.developer.com/deliveryreceipt&senderName=Bob
HTTP/1.1 Host: www.example.com
```

#### Sample response

```
HTTP Status: 201 Created
<SMSResponse version= "1.0" messageid="1234"/>
```

## getSmsDeliveryStatus via GET

When you send an SMS, it is sent through an SMS router (SMSC) in the operator network. This basically queues and then sends your message to the recipient(s). This method lets you know if that message has been delivered.

### Sample request

```
GET </path/>getSmsDeliveryStatus?version=1.0&messageId=1234 HTTP/1.1
Host: www.example.com
```

### Sample response

```
HTTP Status: 200 OK
 ----------HTTP – Body-------------
<DeliveryStatusResponse version="1.0">
   <Message address=tel:+447990123456 status="DeliveredToNetwork"/>
   <Message address=tel:+447990121212 status="MessageWaiting"/>
</DeliveryStatusResponse>
```

## notifySmsDeliveryReceipt via POST

Used for notifying the application when an SMS is delivered to a terminal or if delivery was impossible.

### Sample request

```
POST /notifySmsDeliveryReceipt?correlator=123456 HTTP/1.1
Host: myapp.developer.com

----------HTTP – Body-------------
<deliveryStatus version="1.0">
  <Message address=tel:+447990123456 status="DeliveredToTerminal" />
  <Message address=tel:+447990121212 status="DeliveryImpossible" />
</deliveryStatus>
```

### Sample response

```
HTTP Status: 204 No Content.
```

## getReceivedSms via POST

Enables the application to 'poll' for any SMS received.

### Sample request

```
GET /path/getReceivedSms?version=1.0&registrationId=6789 HTTP/1.1
Host: www.example.com
```

### Sample response

```
HTTP Status: 200 OK
<ReceivedSmsResponse version="1.0">
   <Message content="Vote A" sender="tel:+447790123456" destination="81771" datetime= Tue, 15 Nov 199
   <Message content="Vote B" sender="tel:+447790121212" destination="81771" datetime="Tue, 15 Nov 199
</ReceivedSmsResponse>
```

### startSmsNotification via POST

Used to start the notification of received SMS.

**Sample request**

```
POST /<path>/startSmsNotification?destination=81771&criteria=Vote&
notifyURL= http://myapp.developer.com/voting/&
correlator=123456
HTTP/1.1 Host: www.example.com
```

**Sample response**

```
HTTP Status: 204 No Content
```

### stopSmsNotification via POST

Used by the application to stop the delivery receipt notifications.

**Sample request**

```
DELETE /<path>/stopSmsNotification?correlator=123456 HTTP/1.1
Host: www.example.com
```

**Sample response**

```
HTTP Status: 204 No Content
```

### notifySmsReception via POST

Notifies the application when an SMS is received.

**Sample request**

```
POST /notifySmsReception?correlator=123456 HTTP/1.1
Host: myapp.developer.com
Content-Type: text/xml

<ReceivedSms version="1.0">
<Message content="Vote A" sender="tel:+447790123456" destination="81771" datetime="Tue, 15 Nov
1994 08:12:31 GMT"/>
<Message content="Vote B" sender="tel:+447790121212" destination="81771" datetime="Tue, 15 Nov
1994 08:12:31 GMT"/>
</ReceivedSms>
```

**Sample response**

```
HTTP Status: 204 No Content.
```

## Exceptions

The ONE API widgets can receive exceptions from errors emanating out of Access Gateway mediation primitives as well as out of the service layer.

As part of the exception handling, the ONE API widgets can receive exceptions from two different layers:

- Errors emanating out of Access Gateway mediation primitives
- Errors emanating out of the service layer

All the interfaces may generate the exceptions listed in the following table. Whatever error information is thrown by the mediation flow or service layer is available as part of the error XML tag. In the event that the exception contains messageId and text elements, they are concatenated and presented within this error tag.

*Table 36. Exceptions*

| Exception type | When it is thrown | Exception |
|---|---|---|
| Service | Service-related error has occurred as a result of a client invocation on the service | `HTTP/1.1 500 Internal Server Error`<br>`Content-Type: text/xml`<br><br>`<error>Internal platform error occurred/<err` |
| Service | Operator's service is unavailable | `HTTP/1.1 502 Bad Gateway`<br>`Content-Type: text/xml`<br><br>`<error>Error integrating with the external s` |
| Policy exception | Fault relating to a policy associated with the service | `HTTP/1.1 402 Forbidden`<br>`Content-Type: text/xml`<br><br>`<error>A policy error occurred. Error code i`<br>`Message Length is enforced and message lengt` |
| General | Parameter is specified incorrectly or is missing | `HTTP/1.1 400 Bad Request`<br>`Content-Type: text/xml`<br><br>`<error>Message not present</error>` |
| General | Authorization credentials are incorrect or not present | `HTTP/1.1 401 Unauthorized` |

# Integrating with Web service implementations

Call flow and interface information is included to help you integrate with the Web service implementations that are provided with the Telecom Web Services Server component.

## Integrating with Web service implementations based on SIP/IMS

Telecom Web Services Server supports Web service implementations that operate based on SIP/IMS with Parlay 5.0 Multimedia Messaging (MMM)..

### Integrating with Parlay X Call Notification over SIP/IMS

Parlay X Call Notification over SIP/IMS notifies Web clients of specific call events for a specific called party.

Parlay X Call Notification over SIP/IMS defines interfaces for CallNotificationManager, CallNotification, CallDirectionManager, and CallDirection interfaces. However, this implementation of Parlay X Call Notification over SIP/IMS implements only the **CallNotificationManager** and **CallNotification** interfaces.

Parlay X Call Notification over SIP/IMS is a Parlay X Web services application that provides notification for calls established through the SIP protocol and supports both regular SIP and IMS call flows. This service provides application developers with the ability to handle call notification to other service implementations using simple functions. Through the CallNotificationManager interface of the Parlay X Call Notification over SIP/IMS, the application can register or unregister notifications whenever a network call event occurs for an interested called party. Then, the CallNotification interface is used to provide notification service when a network call event occurs. The type of events that can be reported on by the CallNotification interface include: the called party is on the phone and is busy

(NotifyBusy), the called party can be reached but does not answer the phone (NotifyNoAnswer), or the called party is unreachable (NotifyNotReachable). Parlay X Call Notification over SIP/IMS can also report that a call was attempted (NotifyCalledNumber).

## Call flows

The following is an example call flow showing how the CallNotification interface is invoked. In this example the call flow begins when the SIP Servlet (the part of the code implementing the CallNotification interface) gets a SIP Invite message from the network. The servlet will access the database to get the application addresses that is requesting to be notified for the network call events on the called party. A caller places a call to a callee and the callee is busy (SIP response code 486 Busy Here). The following steps show how the CallNotification would be invoked:

1. The SIP Servlet receives the SIP Invite message from the caller through the network.
2. The SIP Servlet accesses the database to get the list of all applications registered for notifications regarding the callee. Assuming that Application_1 has been registered for the callee, the database returns the search result: Application_1.
3. The SIP Servlet sends an asynchronous notification (NotifyCalledNumber) through PX Notification to Application_1 indicating that a call is attempted for the callee.
4. The SIP Servlet then proxies the Invite message request following the normal call flow to the callee.
5. The SIP Servlet receives 486 (busy) response from the callee through the network.
6. The SIP Servlet sends the busy notification (NotifyBusy) to Application_1.
7. The response follows the normal call flow to the caller.

## Interfaces

**CallNotificationManager**

**startCallNotification**
Enables an application to register a subscription: a request to be notified when a network event occurs for an interested called party.

**stopCallNotification**
Enables an application to unregister an existing subscription.

**CallNotification**

**notifyBusy**
Notifies the registered application for a called party that the called party is busy. The notification is in the form of a network call event notification, as is the case with all operations for the CallNotification interface.

**notifyNoAnswer**
Notifies the registered application that the called party has failed to answer.

**notifyNotReachable**
Notifies the registered application that the called party cannot be reached.

**notifyCalledNumber**
Notifies the registered application that a call was attempted to the callee.

## Integrating with Parlay X Presence over SIP/IMS

By integrating with the Parlay X Presence over SIP/IMS Web service implementation, your client applications can use Web services to subscribe to a presentity, synchronously query the current presence information for a presentity, receive asynchronous notifications about changes in the presence information for a presentity, publish the presence information of a presentity, and unsubscribe from a presentity.

Within an IMS deployment, IBM WebSphere Presence Server interacts with a variety of other presence-savvy endpoints (through the Pw, Pet, Pep, and Peu reference points). As it receives presence information for a particular presentity (an entity whose presence information is of interest), it aggregates the information into a presence document. Integrating with WebSphere Presence Server, the Parlay X Presence over SIP/IMS Web service implementation then maps the IMS Service Control (ISC) presence signaling into a format that is usable by Web service clients.

In response to incoming Web service requests, Parlay X Presence over SIP/IMS sends SIP SUBSCRIBE requests to WebSphere Presence Server. The service implementation processes the corresponding SIP NOTIFY messages from the presence server, and converts the rich presence information into the Parlay X Presence model.

### Call flows

When processing getUserPresence, startPresenceNotification, and endPresenceNotification Parlay X requests, Parlay X Presence over SIP/IMS uses SIP SUBSCRIBE and NOTIFY signaling to communicate with the presence server. SIP SUBSCRIBE and NOTIFY interaction is:

1. An outgoing SUBSCRIBE request to the presence server.
2. An incoming 2xx response to the SUBSCRIBE.
3. An incoming NOTIFY request from the presence server, containing the presence information in the message body.
4. An outgoing 2xx response to the NOTIFY.

When the service processes a getUserPresence request, the outbound SUBSCRIBE message has a SIP Expires header indicating that it is a presence `fetch`. The presence server sends only a single NOTIFY in response.

When the service processes a startPresenceNotification request, the outbound SUBSCRIBE message will have a SIP Expires header indicating that it is an extended subscription. The presence server will send an immediate NOTIFY in response, and may send future NOTIFY messages as the underlying state of the presentity changes. Later, the service may send additional SUBSCRIBE requests to refresh the subscription.

When the service processes an endPresenceNotification request, the outbound SUBSCRIBE message has a SIP Expires header indicating that it wishes to terminate the subscription. The presence server sends only a single NOTIFY in response.

## The PUBLISH operation

When it receives a PUBLISH operation, the Parlay X Presence over SIP/IMS Web service implementation performs the following tasks:

- Verifies that the input is valid and, if it is not valid, returns the appropriate service or exception codes.
- Creates a Presence Information Data Format (PIDF) document. The document contains the body of the PUBLISH request and the Parlay X Presence attributes found in the request, and it specifies a content type of `application/pidf+xml`. (The PIDF format is defined in IETF RFC 3863.)
- Sends the PIDF document to the presence server.

Because the PUBLISH request does not establish a dialog, the response to a PUBLISH request merely indicates whether the request was successful. The presence server returns an exception when the request is not processed successfully; otherwise it returns nothing.

## Interfaces

**PresenceConsumer**

**subscribePresence**
Handles a request to subscribe to the presence of one or more users (presentities). The Web service implementation begins by checking the validity of the requested presentity and uses information available from the Group Resolution mediation primitive to determine whether the presentity is a group URI. The subscribePresence operation then takes a snapshot of the relevant configuration settings as modified by the Web service's SOAP headers. Subsequent Parlay X Presence over SIP/IMS API requests use the same HTTP session and the same configuration settings.

**getUserPresence**
Gets the status of a user (presentity). The Web service implementation verifies that the input is valid. If not, it returns appropriate service exceptions based on incorrect input values. Next, the Web service implementation verifies that the client has previously invoked subscribePresence and been granted authorization for the requested presence attributes. The Web service implementation initiates a SUBSCRIBE/NOTIFY interaction with the presence server to retrieve the relevant presence information and returns the information to the client.

**StartNotification**
The StartNotification request, has a single global transaction ID and its corresponding set of StatusXXX_DeliveryYYY events.

**endNotification**
Terminates a subscription that was created using startNotification.

**startPresenceNotification**
Enables an application to request that it be notified when a presentity's presence information changes. The Web service implementation checks the validity of the requested presentity.

**endPresenceNotification**
Enables an application to request that it no longer be notified when a presentity's presence information changes.

**notifySubscription_DeliveryAttempted**
This marks the attempted delivery of a notification subscription.

**notifySubscription_DeliveryResult**
This marks the delivery result for a single notification subscription.

**subscriptionEnded_DeliveryAttempted**
This marks the delivery result for a subscription ending.

**subscriptionEnded_DeliveryResult**
This marks the delivery result for a single subscription that has ended.

**PresenceSupplier**

**publish**
Allows client applications to publish the presence information for a presentity.

The following PresenceSupplier interfaces are not supported and do not generate usage records:

- getOpenSubscriptions
- updateSubscriptionAuthorization
- getMyWatchers
- getSubscribedAttributes
- blockSubscription

## Integrating with Parlay X Terminal Status over SIP/IMS

Parlay X Terminal Status over SIP/IMS allows client applications to use Web services to synchronously request the status of an IMS terminal (or terminals), and then to receive asynchronous Web service notifications for changes to the state of the terminal. This service also supports group-level operations.

On startup, the IMS terminal for each user must register its existence with an S-CSCF. Because this registration must succeed for the terminal to use any IMS-based services, the existence of this registration indicates if a user is online and reachable, or offline and unreachable. Parlay X Terminal Status over SIP/IMS is used by client applications through Parlay X Web service APIs to determine if an IMS terminal is reachable. To retrieve Parlay X Terminal Status over SIP/IMS from IMS, Parlay X Terminal Status over SIP/IMS behaves as an IMS watcher application on top of the Pw reference point where it retrieves Parlay X Terminal Status over SIP/IMS information from a Presence Server.

### Call flows

When processing getStatus, getStatusForGroup, startNotification and endNotification requests, Parlay X Terminal Status over SIP/IMS uses SIP SUBSCRIBE and NOTIFY signaling to communicate with the presence server. An example of a SIP message flow would be:

1. The Parlay X Terminal Status over SIP/IMS receives an incoming getStatus request for a particular terminal (such as the presentity).
2. The service implementation sends a SIP SUBSCRIBE message to the presence server. This step assumes that the PresenceServerSIPURI is configured.
3. The presence server sends a 200 OK response to the SUBSCRIBE request. The service implementation receives the 200 OK response.
4. The presence server sends a NOTIFY response which includes the presence information of the presentity. The service implementation receives the NOTIFY.

5. The service implementation sends a 200 OK response to the NOTIFY message.
6. The service implementation processes the rich presence information and determines current status of the terminal. It then responds to the incoming getStatus request with the result.

When the Web service implementation processes a getStatus or getStatusForGroup request, the outbound SUBSCRIBE message has a SIP Expires header indicating that it is a presence `fetch`. The presence server then sends only a single NOTIFY in response.

When the service processes a startNotification request, the outbound SUBSCRIBE message has a SIP Expires header indicating that it is an extended subscription. The presence server sends an immediate NOTIFY in response, and may send subsequent NOTIFY messages as the underlying state of the presentity changes. Later, the Web service implementation may send additional SUBSCRIBE requests to refresh the subscription.

When the service processes an endNotification request, the outbound SUBSCRIBE message has a SIP Expires header indicating that it wishes to terminate the subscription. The Web service implementation sends only a single NOTIFY in response.

## Interfaces

**TerminalStatus**

**getStatus**
Obtains the status of the terminal. The Web service implementation generates an outbound SIP SUBSCRIBE request to the configured presence server and uses the incoming presence information to determine whether the terminal is busy, reachable, or unreachable.

**getStatusForGroup**
Performs the same query as getStatus, except that it accepts group lists and can return multiple results simultaneously.

**TerminalStatusNotificationManager**

**startNotification**
Like getStatus and getStatusForGroup, this operation uses SIP SUBSCRIBE and NOTIFY interactions to retrieve presence information. However, startNotification differs in that it creates long-lived subscriptions instead of performing a single fetch operation. The Web service implementation then uses the PX Notification component Web service to deliver Web service callbacks to a Web service endpoint that is implementing the TerminalNotification interface.

**endNotification**
Terminates a subscription that was created using startNotification.

**statusNotification_DeliveryAttempted**
The attempted delivery of a StatusNotification.

**statusNotification_DeliveryResult**
The delivery result for a single StatusNotification.

**statusChanged_DeliveryAttempted**
The attempted delivery to change the status.

**statusChanged_DeliveryResult**
The result of a single change in status.

**statusError_DeliveryAttempted**
The attempted delivery of a StatusError.

**statusError_DeliveryResult**
The delivery result for a single StatusError.

**statusEnd_DeliveryAttempted**
The attempted delivery to end the status.

**statusEnd_DeliveryResult**
The delivery result for a single status to end.

## Integrating with Call Handling over SIP/IMS

The Call Handling over SIP/IMS Web service provides a mechanism for an application to designate how calls are to be handled for a specific number.

Call Handling over SIP/IMS provides a rule-based processing capability that applications can access through the following set of operations: setRules, getRules, setRulesForGroup, and clearRules.

The Call Handling over SIP/IMS Web service is designed to provide a simplistic method for applications, at they relate to call behavior. The interface functions on a request/response message from the Web client to a Web service. When actual calls are routed, the Web service validates whether the calls are permitted according to a configured address.

### Call flows

The following is an example call flow showing how the Call Handling over SIP/IMS interfaces are invoked. In this example the call flow begins when the application client invokes the Web service by routing the SOAP message over the HTTP to the Telecom Web Services Access Gateway. The ESB acts as a Web service between the client application and the Call Handling service. The following steps show how the Call Handling over SIP/IMS is invoked:

1. You attempt to make a call from a Parlay gateway.
2. CallHandlingNotificationManager receives a start/stop event from the Parlay Connector.
3. CallHandlingRuleProcessor evaluates the rules, and the CHRule executes the rules.
4. The call negotiation is complete, and you can now start your phone conversation.

### Interfaces

**setRules**
Sets the Call Handling rules for the call destination address. If a set of rules is already in place for the address, then this operation will replace the old rules with the set provided in the operation. The address may not specify a group. If a group is specified, a PolicyException will return.

**setRulesForGroup**
Sets the Call Handling rules for multiple call destinations. If a set of rules are already in place for any of the addresses, then this operation will replace the old rules with the set provided in the operation. The addresses

may include groups with arbitrary prefix for group which will be resolved in the Telecom Web Services Access Gateway for members that use the *tel*:, *sip*:, and *sips*: URIs.

**getRules**
Retrieves the Call Handling rules for the call destination. The address may not specify a group. If a group is specified, a PolicyException will return.

**clearRules**
Clears the Call Handling rules associated with the specified address. If rules have not been set for an address, then the operation silently ignores the request and does not return a fault message. The addresses may include groups with members using the *tel*: and *sip*: URIs.

**Note:** Wildcards cannot be used to specify addresses.

## Integrating with Parlay X Third Party Call over SIP/IMS

Parlay X Third Party Call over SIP/IMS provides the ability to initiate a call from a network entity between two different users or user agents.

Parlay X Third Party Call over SIP/IMS provides simple (high level) Parlay X functions to application developers that can be used to determine how to initiate a third party call. Using this Web service application developers can quickly develop applications use without having detailed knowledge of the telecommunications field. Parlay X Third Party Call over SIP/IMS works on a request or response message from a client to a Web service application. The implementation includes SIP Application and SIP Servlets that are required to run in a WebSphere converged HTTP and SIP container. The call initiation is done through a SOAP over HTTP request.

Parlay X Third Party Call over SIP/IMS provides Parlay X Web service support for applications with the ability to initiate a call between two addresses (makeCall), retrieve the current call status after a call has been initiated (getCallInformation), terminate a call (endCall), or cancel a call if the parties have not yet been reached (cancelCall).

### Call flows

This is a sample call flow for the makeCall Webservice operation:

1. The application client invokes a Web service with two user endpoints (*User Agent 1* and *User Agent 2*) using SOAP message over HTTP to Access Gateway.
2. If there is a service policy for Parlay X Third Party Call over SIP/IMS then Access Gateway will send this information through SOAP message to the service.
3. Access Gateway invokes Parlay X Third Party Call over SIP/IMS, with any policy information as discussed in Step 2.
4. Parlay X Third Party Call over SIP/IMS will contact the Admission control service to ensure the service does not exceed the server or cluster capacity.
5. The common service Privacy is called to ensure that the requester is allowed to view requested information.
6. Traffic Shaping component Web service is called to control the rate of traffic (messages) to the element in the network. The traffic estimate size can be configured for the method being called using the trafficEstimatePerRequest property. A default 50 to 1 ratio means for every Parlay X request there will be up to 50 SIP requests generated. When planning, along with figuring the

number of outstanding SIP requests that will be supported, the duration of the call needs to be taken into consideration.

7. The Parlay X Third Party Call over SIP/IMS creates a new SIP application session, with the initial state `Call_Initial`.
8. The SIP call signals now flow from the SIP Servlet 1 to the Out Bound Proxy address, which can be referred to also as the S-CSCF.
9. Call Information data is stored in the database table THIRDPARTYCALL.
10. SIP call negotiation between Servlet 1 call flow and User Agents is taking place.
11. The call negotiation is completed and the User Agents can now start their phone conversation through the established Mobile media connection.

## Interfaces

### ThirdPartyCall

**makeCall**
Initiates a call between two addresses, CallingParty and CalledParty. Optionally the application can also indicate the charging information (Charging). This is determined by the policy information configured per provider.

**getCallInformation**
Retrieves the current call status of the CallIdentifier (a parameter returned from makeCall). A CallInformation structure is returned, containing StartTime, Call status, Duration, and TerminationCause (if applicable). Provided that the call status record has not expired and has not been purged, this method can be invoked multiple times even if the call has ended.

**endCall**
Terminates the call identified by CallIdentifier.

**cancelCall**
Cancels the call identified by CallIdentifier. This operation has no effect if a call has been connected.

## Session affinity

Session affinity is provided through WebSphere Application Server. To make use of this capability for session affinity, the calling Web service client must preserve the same HTTP session for subsequent calls related to a corresponding SIP session.

In your Third Party Call application, therefore, requests that refer to the result of a previous makecall invocation must preserve the HTTP session. A Java application would preserve the HTTP session by setting SESSION_MAINTAIN_PROPERTY in the Call class.

Here is an example of using the Java Call class to preserve the HTTP session:

```
ServiceFactory factory = ServiceFactory.newInstance();
Service service = factory.createService(new Qname("http://www.provider.ParlayX..."));
Call call = service.createCall();
:
call.setProperty(call.SESSION_MAINTAIN_PROPERTY,new Boolean(true));
:
// rest of class
```

# Integrating with Web service implementations based on IBM WebSphere software

Telecom Web Services Server (TWSS) supports Web service implementations that interoperate with other components within the IBM WebSphere product group–including WebSphere Application Server, IBM XDMS, and other instances of TWSS.

## Integrating with Parlay X Payment (PostPaid)

Parlay X Payment (PostPaid) allows you to easily enable an application to charge an amount against a user account or place money into a user account. The Web service then writes this charging information to a local database.

Parlay X Payment (PostPaid) is designed as a simple Parlay X Web services support API to enable billing support through a Web interface. Using this high-level abstraction Parlay X specification, developers can quickly create payment applications to interact with Parlay X Web services APIs though a simple XML-based messaging exchange.

### Call flows

The following steps show the general process flow for this Web service using an offline charging system:

1. A Parlay X payment client (an application created to interface with Parlay X Payment (PostPaid)) sends charging (billing) information in a SOAP message over HTTP to Access Gateway.
2. Access Gateway inspects the Web service request and adds policy data retrieved from Subscription Management to the SOAP message.
3. Access Gateway invokes Parlay X Payment (PostPaid).
4. Parlay X Payment (PostPaid) invokes Admission Control component Web service to determine if the system has enough capacity to handle the request. If it does process continues, if not an error is generated and returned to the payment client.
5. If processing continues, the TWSS Administration Console obtains the policy data from the SOAP message. If the policy data is missing, the policy defaults that were set through the TWSS Administration Console.
6. If Parlay X Payment (PostPaid) has Payment Usage Records enabled, payment charging data is sent to Usage Record component Web service, and a record is created in the service usage database.
7. If Parlay X Payment (PostPaid) has CEI enabled, Parlay X Payment (PostPaid) creates a common base event representing a payment record in the CEI event repository.
8. If Parlay X Payment (PostPaid) has Results Usage Records enabled, payment service results data is sent to Usage Record component Web service, and a record is created in the service usage database.
9. The billing mediator application (an application created to interface with Parlay X Payment (PostPaid)) pulls the payment record from the service usage database and the CEI event repository and updates the user account information on the database in the billing domain.
10. The billing mediator application may delete the payment record from the service usage database and the CEI event repository when the user account is successfully updated.

11. If a fault or alarm condition occurs during the process flow Parlay X Payment (PostPaid) sends error information to Fault and Alarm component Web service, which creates fault and alarm records (as common base events) in the CEI repository and outputs alarms to JMX.

### Interfaces

**AmountCharging**

**chargeAmount**
Charge an amount against a user account.

**refundAmount**
Place money into a user account.

**VolumeCharging**

**chargeVolume**
Charge against a user account an amount that has been specified as a volume.

**refundVolume**
Place into a user account money that has been specified as a volume.

## Integrating with Parlay X Address List Manager over XCAP

The Parlay X Address List Manager over XCAP Web service provides Web service-based mechanisms to create, read, update, and delete group definitions stored within the XDMS. This service uses the XCAP protocol rather than just the group definitions. This service implementation interfaces with XDMS which provides an XML Configuration Access Protocol (XCAP).

The Parlay X Address List Manager over XCAP Web service defines the faults or alarms that are replaced. All faults or alarms are replaced by the component name and the operation name.

### Error handling

The following is an example of error handling, which displays the HTTP response codes that are generated and returned by the XDMS system. Additional HTTP response codes can be returned by the HTTP Server or WebSphere Application Server.

- 200 OK – A GET, DELETE, or PUT which performed an Update completed successfully.
- 201 Created – The document, element, or XML attribute was created successfully.
- 304 Not Modified – The Etag of requested document matches that specified in the If-None-Match header.
- 400 Bad Request- Name Space not included in the query component of the request.
- 403 Forbidden – Used to indicate that authorization has failed
- 404 Not Found – The request is for an AUID that the XDMS does not support, or a GET or DELETE is for an item that does not exist in the XDMS.
- 405 Method Not Allowed - Post is not supported.
- 409 Conflict - Should return XML.
- 412 Precondition Failed – See the use of If-Match, and If-None-Match headers.

- 415 Unsupported Media Type – If a PUT request contains an element and Content-Type is not ″application/xcap-el+xml″, or a PUT request contains an attribute and Content-Type is not ″application/xcap-att+xml″.
- 500 Internal Error – Used to indicate that an error occurred in the XDMS that has nothing to do with the request.

The Address List Manager handles response codes as follows:

- Response codes between 100 and 199: the response is ignored.
- Response codes between 200 and 299: a Parlay X output message is returned.
- Response codes 304, 400, 403, 404, and 415: SVC0002 is returned.
- All other response codes: SVC0001 is returned.

When either SVC0001 or SVC0002 is returned, the response code is included in the message.

### HTTP response codes

*Table 37. SOAX faults caused by HTTP response codes*

| HTTP Response code | SOAX number |
|---|---|
| 304 | 6020 |
| 400 | 6021 |
| 403 | 6022 |
| 404 | 6023 |
| 405 | 6024 |
| 409 | 6025 |
| 412 | 6026 |
| 415 | 6027 |
| 500 | 6028 |
| Other 300-999 responses | 6029 |

### Group URI mapping

The Address List Manager URI maps the group URIs that are used in the Parlay X Address List Manager over XCAP, and the IBM XDMS XCAP URI.

The following is the Address List Manager group URI format:

```
GroupScheme:list_name@user_domain.provider_domain;user=user_uri
```

where:

**GroupScheme**
- Specified for the MBean property.
- Determines whether a URI is a group URI or an XCAP User Id (XUI).
- The value of a scheme MBean property should not be the same as the scheme used for user URIs.

**list_name**
Group list name

**user_domain**
The domain specified by or for the user (specified in the Parlay X API parameter).

**provider_domain**
The domain specified in the Mbean property. Should not be passed as part of the parameters on the Parlay X Address List Manager over XCAP API.

**user_uri**
- A URI that corresponds to any user, for example *sip:user@ibm.com*.
- Constant global for global groups.

For example, in the URI `group:mybuddies@friends.ibm.com;sip:user@ibm.com`:

The group scheme is `group`.

The group list name is `mybuddies`.

The subdomain is `friends`.

The service provider domain is `ibm.com`.

The user is `sip:user@ibm.com`.

Here are some examples of XCAP URIs:

**User Group**
`XCAP RootURI/AUID/users/user_uri/file_name/~~/Node_Selector`

**Global Group**
`XCAP RootURI/AUID/global/file_name/~~/Node_Selector`

**Mapped XCAP URI**
`http://xdms.example.com:9080/services/resource-lists/users/sip:user@example.com/friends`

Each of these example URIs contains some or all of the following elements:

**XCAP Root URI**
The context root for the application, for example `http://xdms.ibm.com:9080/services/`.

**AUID** An XCAP application-unique ID that defines the type of XML document. AUIDs are defined according to XCAP usage specifications, for example resource-lists or rls-services.

In Parlay X Address List Manager over XCAP, the AUID value is always resource-lists.

**Document Selector**
Identifies the specific document to be stored or accessed. XCAP documents are segregated into two tree branches:
- Global: Contains documents which can only be created by administrators, but accessed by anybody.
- User: Contains user-specific documents for a specific XCAP User Identifier (XUI).

In both branches, a `file_name` is a user-selected name for the group for example, `friends`. It is specified on the Parlay X Address List Manager over XCAP MBean configuration.

Examples:

`user/file_name`

`global/file_name`

**Node Selector**
An optional XPATH expression that can be used to identify a specific XML

element or attribute which is to be updated, retrieved, or deleted. The `/~~/` is a separator between the document selector and the node selector.

## Interfaces

### GroupManagement

**createGroup**
Creates a group definition.

If the autoName parameter is set to `true`, the Address List Manager attempts to create unique groups if the original name already exists, and then sends the request.

IBM XDMS supports autoName. To take advantage of this support, open the Integrated Solutions Console and update the ibm-xdms Resource Environment Provider by setting enableDraftPutConditionalSupport to `true`.

**deleteGroup**
Deletes a group definition.

### Group

**addMember**
Adds a member to the specified group.

**addMembers**
Adds multiple members to the specified group.

**deleteMember**
Removes a member from the specified group.

**deleteMembers**
Removes multiple members from the specified group.

**queryMembers**
Retrieves the names of the members of the specified group.

**addGroupAttribute**
Adds a new attribute or updates an existing attribute for the specified group.

**deleteGroupAttribute**
Deletes an attribute from the specified group.

**queryGroupAttributes**
Retrieves a list of attributes, and their values, for the specified group.

**addGroupMemberAttribute**
Adds a new attribute or updates an existing attribute for the specified group member.

**deleteGroupMemberAttribute**
Deletes an attribute for the specified group member.

**queryGroupMemberAttributes**
Retrieves a list of attributes, and their values, for the specified group member.

# Integrating with Web service implementations based on Direct Connect protocols

Telecom Web Services Server supports Web service implementations which operate on direct connect protocols.

## Integrating with WAP Push over SMPP

The WAP Push over SMPP Web service implementation provides a Web service that accepts requests for sending WAP Push messages, that control various checks which are done to ensure that the requests are valid. It will then forward the requests to the appropriate network elements for downstream processing. The WAP Push service implementation will then forward the delivery receipt notifications to the appropriate sender applications.

### WAP Push over SMPP asynchronous call flow

The following steps show an example of the Asynchronous sendWAPPushSI call flow.

1. The client application invokes a sendWAPPushSI operation.
2. The request will be queued in the JMS Que for further processing.
3. WAP Push over SMPP sends the WAPPush message request through the JCA adapter to the SMSC using a *submit_sm* (for a single target) or *submit_sm_multi* (for multiple targets) PDUs.
4. The SMSC returns a *submit_sm_resp* or *submit_sm_multi_resp* PDU response through the SMPP JCA adapter to the WAP Push over SMPP service.
5. The WAP Push over SMPP service returns a Transaction Id to the client application.
6. The SMSC then forwards the WAPPush message to the targeted mobile device.
7. If the confirm delivery option is opted, a *delivery_sm* PDU will be sent by the SMSC with a final delivery status through the JCA adapter to the WAP Push over SMPP.
8. The WAP Push over SMPP service invokes a *notifyWAPPushDeliveryReceipt* Web service call on the notification client application.

### WAP Push over SMPP synchronous call flow

The following steps show an example of the Synchronous sendWAPPushSI call flow.

1. The client application invokes a sendWAPPushSI operation.
2. The WAP Push over SMPP sends the WAPPush message request through the JCA adapter to the SMSC using *submit_sm* (for a single target) or *submit_sm_multi* (for multiple targets) PDUs.
3. The SMSC returns a *submit_sm_resp* or *submit_sm_multi_resp* PDU response through the SMPP JCA adapter to the WAP Push over SMPP service.
4. The WAP Push over SMPP service returns a network Id as response to the client.
5. The SMSC then forwards the WAPPush message to the targeted mobile device.

### Interfaces

**SendWAPPush**

**Note:** The SendWAPPush interface is provided as a Web service.

**sendWAPPushSI**
Sends a WAP Push message request.

**sendWAPPushSL**
Not supported.

**sendWAPPushCO**
Not supported.

**getWAPPushDeliveryStatus**
Retrieves the status request for the previous WAP Push delivery.

**WAPPushNotification**

**Note:** The WAPPushNotification interface is used as a client only.

**notifyWAPPushDeliveryReceipt**
Notifies the application when a WAP Push message is delivered, when a message is delivered to the terminal of the recipient, or when delivery is impossible.

## Integrating with Parlay X SMS over SMPP

The Parlay X SMS over SMPP Web service implementation provides operations for sending an SMS message to the network, monitoring the delivery status of a sent SMS message, and asynchronously receiving notification of message delivery status.

### Call flows

The following steps show an example of the call flow for Parlay X SMS over SMPP:

1. The application invokes a sendSMS.
2. Parlay X SMS over SMPP sends the sendSMS through the JCA adapter to the SMSC.
3. The SMSC sends the request to the appropriate server.
4. The SMSC returns an SmsNotification response through the JCA adapter to Parlay X SMS over SMPP.
5. Parlay X SMS over SMPP sends the SmsNotification response to the application.

### Interfaces

**Note:** These interfaces are provided as Web services–except for SmsNotification, which is used as a client only.

**SendSms**

**sendSms**
Allows you to send an SMS request and monitor the status of that request.

**sendSmsLogo**
Allows you to request the sending of an SMS logo to a specified address or address set, specified as destinationAddressSet. Optionally, the application can also specify the sender name that is displayed on the user's terminal (senderName) and the name of an operator-specific charging plan (charging).

**sendSmsRingtone**

Allows you to request the sending of an SMS ring tone, specified by the string ringtone (in RTX format), to a specified address or address set, specified as destinationAddressSet. Optionally, the application can also specify the sender name that is displayed on the user's terminal (senderName) and the name of an operator-specific charging plan (charging).

**getSmsDeliveryStatus**

Requests the status of a previous SMS delivery request identified by requestIdentifier. The information on the status is returned in deliveryStatus, which is an array of status related to the request identified by requestIdentifier. The status is identified by a couplet indicating a user address and the associated delivery status.

**SmsNotification**

**notifySmsReception**

Requests the status of a previous SMS delivery request identified by requestIdentifier. The information on the status is returned in deliveryStatus, which is an array of status related to the request identified by requestIdentifier.

**notifySmsDeliveryReceipt**

When an SMS message is sent to multiple terminals, requests a notification from each terminal. The notification indicates one of three outcomes: delivery was successful, time expired before the message could be delivered, and notification is not supported for the terminal.

**ReceiveSms**

**getReceivedSms**

Retrieves all of the SMS messages received that fulfill the criteria identified by registrationIdentifier. The method returns only the list of SMS messages received since the last time the method was invoked. (Older messages are removed from the server.)

**SmsNotificationManager**

**startSmsNotification**

Enables an application to request notifications for short messages online.

**stopSmsNotification**

Enables an application to stop receiving notifications for short messages online.

## Usage records

The interfaces for Parlay X SMS over SMPP generate usage records as follows.

| Interface | Usage records written |
|---|---|
| SendSms: sendSms | 1 usage record per target per operation, without delivery confirmation |
| SendSms: sendSmsLogo | 1 usage record per target per operation, without delivery confirmation |
| SendSms: sendSmsRingtone | 1 usage record per target per operation, without delivery confirmation |

| Interface | Usage records written |
|---|---|
| SendSms: getSmsDeliveryStatus | 1 usage record per operation |
| SmsNotification: notifySmsReception | None |
| SmsNotification: notifySmsDeliveryReceipt | 1 usage record per target per SendSms operation, written after delivery confirmation is received |
| ReceiveSms: getReceivedSms | 1 usage record per operation |
| SmsNotificationManager: startSmsNotification | 1 usage record per operation |
| SmsNotificationManager: stopSmsNotification | 1 usage record per operation |

For more details, refer to the topic *Usage records for Parlay X SMS over SMPP*.

### SMPP operations

The SMPP JCA adapter supports the following SMPP operations:

**bind_transceiver**: The SMPP supports the bind_transceiver operation, with which an ESME can bind to an SMSC as a transceiver (called an ESME transceiver).

**bind_transmitter**: The SMPP supports the bind_transmitter operation, with which an ESME can bind to an SMSC as a transmitter (called an ESME transmitter).

**bind_receiver**: The SMPP supports the bind_receiver operation, with which an ESME can bind to an SMSC as a receiver (called an ESME receiver).

**unbind**: Registers an ESME instance of SMSC, which informs the SMSC of the discontinued use of the network connection for the submission of delivery messages.

**generic_nack**: A generic negative acknowledgement to an SMPP PDU submitted with an incorrect message header.

**submit_sm**: Uses the ESME to submit short messages to the SMSC for onward transmissions of a specified SME. This operation does not support the transaction message mode.

**submit_multi** : Can be used to submit an SMPP message for the delivery of one or multiple recipients to a distribution list. This operation does not support the transaction message mode.

**deliver_sm**: Issued by the SMSC to send a message to an ESME. Using this command, the SMSC can route a short message to the ESME for delivery confirmation.

**enquire_link**: A message that can be sent by either an ESME or an SMSC to provide a confidence check of the communication path between the ESME and the SMSC. On receipt of this request, the receiving party should respond with an enquire_link_resp. This verifies that the application-level connection between the ESME and the SMSC is functioning.

## Integrating with Parlay X Terminal Location over MLP

The Parlay X Terminal Location over MLP describes how as part of the implementation each individual Web service operation will be mapped to the interfaces provided by the MLP Protocol.

## Call flows

Call flows are provided for the following request and response major code paths:

- Maps incoming Parlay X parameters to MLP parameters.
- Maps the response MLP values to Parlay X understandable parameters.

The following is an example call flow showing how calls to the Service Platform components are invoked. In this example the call flow begins when the Parlay X 2.1 Terminal Location Requests and the corresponding HTTP requests to the backend Location server are synchronous operations:

1. The Web service client request is received by the TWSS Access Gateway.
2. The SOAP request is forwarded to the Web service implementation.
3. The Web service implementation receives the request and validates the input parameters. It then converts the input parameters from Parlay X data structures to parameters that are understandable by MLP.
4. An HTTP URL connection with the MLP server is opened. The connection is used for both request and response flows.

   **Note:** For triggered notification operations, the notification data will be stored in a database table and updated when changes take place.
5. The HTTP URL connection receives the response data. A Callback Servlet listens for the triggered response messages for notification requests.
6. The retrieved response objects (Response and Error) are converted to Parlay X data structures.
7. The response from the location-based synchronous operation is forwarded to the Access Gateway, which passes the response to the Web service client.
8. The triggered notification operation sends the response/notification information to the notification endpoint.
9. The database table is updated with the current status.

The HTTP connections are used for both the request and response flows. All Terminal Location notification requests are intended to be triggered by an event or based on a periodic timer. The callback responses are received asynchronously from the MLP location server and are then processed by a Callback Servlet in order to deliver the notification to a third-party application.

## Interfaces

**TerminalLocation**

**getLocation**
Retrieves the location of a single terminal. The accuracy requested is the desired accuracy for the response. The acceptable accuracy is the limit acceptable to the requester. The URI provided is for a single terminal.

**getLocationForGroup**
Retrieves the location of a group of terminals. Its function is the same as GetLocation, except that a group URI is expected as input.

**getTerminalDistance**
Determines the distance of a single terminal from a location. The URI provided cannot be a group URI.

**TerminalLocationNotificationManager**

**startGeographicalNotification**

Registers an application server to receive a notification when a terminal enters or exits an area specified by a longitude, latitude, and radius.

**Note:** This implementation is provided only when the MLP 3.2 specification is selected. If MLP 3.1 is selected as the default, the request is ignored and a corresponding error message is returned to the Web service client.

**Note:** MLP provides the following format for specifying both duration and frequency: *ddhhmmss*. As a result, the maximum possible value is 99 days, 23 hours, 59 minutes, and 59 seconds.

**startPeriodicNotification**

Registers an application server to receive a notification at some given interval.

**Note:** This implementation is provided for both MLP 3.1 and MLP 3.2 specifications.

**Note:** MLP provides the following format for specifying both duration and frequency: *ddhhmmss*. As a result, the maximum possible value is 99 days, 23 hours, 59 minutes, and 59 seconds.

**endNotification**

De-registers an application server from receiving notifications.

**TerminalLocationNotification**

**locationNotification**

Sent to an application when a notification is received.

**locationError**

Sent to an application when a notification error occurs.

**locationEnd**

Indicates that the notifications have ended for this application; called when the duration or count for notifications has been completed.

## Supported MLP shapes

Versions 3.1 and 3.2 of the MLP protocol define various shapes for representing geographical areas in which a mobile subscriber can be located. The following shapes are supported by this Web service implementation:

- Point
- CircularArea
- CircularArcArea
- EllipticalArea
- LineString
- LinearRing
- Box
- Polygon
- MultiPoint

Refer to the MLP specification for the precise definitions of these shapes.

### Usage records

The interfaces for Parlay X Terminal Location over MLP generate usage records as follows.

| Interface | Usage records written |
|---|---|
| TerminalLocation: getLocation | 1 usage record per operation |
| TerminalLocation: getLocationForGroup | 1 usage record per URI target in the group |
| TerminalLocation: getTerminalDistance | 1 usage record per target |
| TerminalLocationNotificationManager: startGeographicalNotification | 1 usage record per target |
| TerminalLocationNotificationManager: startPeriodicNotification | 1 usage record per target |
| TerminalLocationNotificationManager: endNotification | 1 usage record per operation |
| TerminalLocationNotification: locationNotification | None |
| TerminalLocationNotification: locationError | None |
| TerminalLocationNotification: locationEnd | None |

For more details, refer to the topic *Usage records for Parlay X Terminal Location over MLP*.

## Integrating with Parlay X Multimedia Messaging over MM7

The Parlay X Multimedia Messaging over MM7 Web service sends a message to an MMSC (Multimedia Message Service Center) using the MM7 interface for onward transmission. The Mm7Callback MDB (Message Driven Bean) performs this operation when called with the onMessage ( ) method, and when the message is sent from the MM7 Protocol Converter.

The Parlay X Multimedia Messaging over MM7 Web service defines interfaces for MessageNotification, SendMessage, ReceiveMessage, and MmsNotification Manager. The MMS message to be exchanged between client and server must be in the form of one or more attachments with appropriate content. This is defined by SOAP messages with attachments.

The Telecom Web Services Access Gateway supports flow-through of SOAP attachments from the Access Gateway to the TWSS Web service implementation. It can be configured to pass through the attachments or to be discarded before invoking service implementations. It requires the retrieval and storage of SOAP attachments in a MMS send path, and the retrieval and attachment during the MMS message retrieval. The MMS Web service implementation remains the same for both approaches.

### Call flows

The following are examples of call flows showing how the SendMessage interface is invoked using Synchronous and Asynchronous mode. In these examples, the call flow begins when the Access Gateway is configured to pass through the SOAP attachments to TWSS.

**Asynchronous mode**

1. The Data Handler is configured to act before the invocation of SendMessage.
2. The Data Handler retrieves the SOAP attachments, then stores the attachments in an Attachment content table and the metadata in a metadata table.
3. The Attachments are removed from the original SOAP message.
4. The SendMessage operation is invoked.
5. The control flows to the Attachment Layer.
6. The Attachment Layer retrieves the attachments and passes them to the connector.

**Synchronous mode**

1. The Data Handler is configured to act before the invocation of SendMessage.
2. The Data Handler, on identifying the mode of operation as Synchronous, performs no further processing and passes control to the next layer.
3. The SendMessage operation is invoked.
4. The control flows to the Attachment Layer.
5. The Attachment Layer retrieves the attachments from the message context and passes them to the connector.

The following is an example call flow showing how the ReceiveMessage interface is invoked. In this example, the call flow begins when the Access Gateway is configured to pass through the SOAP attachments to TWSS.

1. The Attachment layer receives the MMS message containing attachments.
2. The Attachment layer retrieves the attachments and stores them in the attachment database.
3. The Client is notified of reception when the MMS message invokes notifyMessageReception and passes.
4. The Client invokes getMessage to retrieve the MMS message.
5. The getMessage interface marks the expiration time to the current time in the Receive Data Table.
6. The Data Handler is configured to act after the invocation of getMessage.
7. The Data Handler retrieves the attachments and metadata from the attachment database, then adds them as SOAP attachments.
8. The attachments and metadata are deleted from the attachment database.

## Interfaces

**Note:** These interfaces are provided as Web services–except for MessageNotification, which is used as a client only.

**SendMessage**

**sendMessage**
Submits a request to send an MMS message.

**getMessageDeliveryStatus**
Retrieves the status of a specified previous send request.

**ReceiveMessage**

**getReceivedMessages**
Retrieves all of the MMS messages having a specified correlator.

**getMessage**
Retrieves the entire MMS message identified by `messageRefIdentifier`.

**getMessageURIs**
Not supported.

**MessageNotificationManager**

**startMessageNotification**
Initiates notifications to the application for a specified MMS service activation number and criterion.

**stopMessageNotification**
Ends notifications to the application for a specific startMessageNotification operation.

**MessageNotification**

**notifyMessageReception**
Notifies the application when an MMS message sent by the application has been received.

**notifyMessageDeliveryReceipt**
Notifies the application of the disposition of an MMS message sent by the application: the message was delivered successfully, the message could not be delivered before time expired, or notification is not supported for one or more of the recipient addresses.

### Usage records

The interfaces for Parlay X Multimedia Messaging over MM7 generate usage records as follows.

| Interface | Usage records written |
|---|---|
| SendMessage: sendMessage | 1 usage record per target without delivery confirmation |
| SendMessage: getMessageDeliveryStatus | 1 usage record per operation |
| ReceiveMessage: getReceivedMessages | 1 usage record per operation |
| ReceiveMessage: getMessage | 1 usage record per operation |
| ReceiveMessage: getMessageURIs | None |
| MessageNotificationManager: startMessageNotification | 1 usage record per operation |
| MessageNotificationManager: stopMessageNotification | 1 usage record per operation |
| MessageNotification: notifyMessageReception | None |
| MessageNotification: notifyMessageDeliveryReceipt | 1 usage record per target |

For more details, refer to the topic *Usage records for Parlay X Multimedia Messaging over MM7*.

## Integrating with Parlay-based Web service implementations

Telecom Web Services Server supports Web service implementations that operate over Parlay 3.x and 4.x APIs with Parlay 5.0 Multimedia Messaging (MMM).

## Integrating with Parlay X Terminal Status over Parlay

Operations of the Parlay X Terminal Status over Parlay Web service operation map to Parlay Mobility Service Capability Feature (SCF) API methods.

### Call flows

The following steps show an example of the call flow for Parlay X getStatus () Web service request:

1. The application invokes a getStatus request.
2. The Parlay X Terminal Status over Parlay Web service implementation sends the request over the Parlay API to the Parlay Connector.
3. The Parlay Connector sends the request to the appropriate server and waits for a response.
4. The server sends a GetStatusResponse to the Parlay Connector.
5. The Parlay Connector sends the GetStatusResponse to Parlay X Terminal Status over Parlay.
6. Parlay X Terminal Status over Parlay sends the response to the application.

### Interfaces

**Terminal Status**

**getStatus**
Retrieves the status for a single terminal. It is mapped to the IpUserStatus.statusReportReq Parlay Mobility SCF API function. It is also used by applications that are interested in determining the status of one or more terminal devices. When status information is available, the IpAppUserStatus.statusReportRes callback is invoked. If an error occurs, the IpAppUserStatus.statusReportErr callback is invoked.

**getStatusForGroup**
Initiates a retrieval activity, where one or more terminals or groups of terminals, may have their statuses determined. The Web service may return a result set that does not include complete information, allowing the Web service implementation to choose to deliver a partial set of results that accommodate other conditions. This includes avoiding time-outs. In this case, the addresses for which no attempt has been made to provide data, will be marked NotRetrieved in the result for each applied address.

**Terminal Status Notification Manager**

**startNotification**
Applications can be notified of status changes, and the number and duration of notifications can be requested as part of the notification setup. The notification may also be governed by service policies, or a combination of both. The StartNotification will be mapped to the IpUserStatus.triggeredStatusReportingStartReq Parlay Mobility SCF API function.

**endNotification**
The application may end a notification using this operation, which the service will map to IpUserStatus.triggeredStatusReportingStop. However until this operation returns, the notifications may continue to be received by the application. There is not an end of

notification (statusEnd) message delivered to the application for a notification ending using this operation.

**Terminal Notification**

**statusNotification**
When the status of monitored devices change, a notification will be delivered to the application with the new status information for each of the devices. If a group identifier was used the terminal device URI will be provided, and not the group URI. The sending of this notification occurs when a service callback is invoked by the following Parlay API functions.

**statusError**
The status changed error message will be sent to the application to indicate that the notification is being cancelled by the Web service. The sending of this notification occurs when a service callback is invoked by the following Parlay API functions:

- IpAppUserStatus.triggeredStatusReportErr
- IpAppUserStatus.StatusReportErr (Valid only when checkImmediate is indicated).

**statusEnd**
This message will be delivered when the duration or count for notifications have completed. The message will not be delivered in the case of an error ending notification or deliberate ending of the notification. This is indicated when the application invokes the TerminalStatusNotificationManager.endNotification operation.

## Integrating with Parlay X Terminal Location over Parlay

The Parlay X Terminal Location over Parlay Web service implementation provides operations for sending Terminal Location requests and registering for Terminal Location notifications. This service implementation interacts with a Parlay gateway using a Parlay Connector.

### Call flows

The following steps show an example of the call flow for Parlay X Terminal Location over Parlay:

1. The application invokes a getLocation request.
2. Parlay X Terminal Location over Parlay sends the request through the Parlay Connector to the Parlay gateway.
3. The Parlay gateway sends the request to the appropriate server and waits for a response.
4. The server sends a LocationInfo response to the SMSC.
5. The Parlay gateway returns the response through the Parlay Connector to Parlay X Terminal Location over Parlay.
6. Parlay X Terminal Location over Parlay sends the response to the application.

### Interfaces

**TerminalLocation**

**getLocation**
Retrieves the location of a single terminal. The accuracy requested

is the desired accuracy for the response. The acceptable accuracy is the limit acceptable to the requester. The URI provided is for a single terminal.

**getLocationForGroup**
Retrieves the location of a group of terminals. Its function is the same as GetLocation, except that a group URI is expected as input.

**getTerminalDistance**
Determines the distance of a single terminal from a location. The URI provided cannot be a group URI.

**TerminalLocationNotificationManager**

**startGeographicalNotification**
Registers an application server to receive a notification when a terminal enters or exits an area specified by a longitude, latitude, and radius.

**startPeriodicNotification**
Registers an application server to receive a notification at some given interval.

**endNotification**
De-registers an application server from receiving notifications.

**TerminalLocationNotification**

**locationNotification**
Sent to an application when a notification is received.

**locationError**
Sent to an application when a notification error occurs.

**locationEnd**
Indicates that the notifications have ended for this application; called when the duration or count for notifications has been completed.

## Integrating with Parlay X SMS over Parlay

The Parlay X SMS over Parlay Web service implementation provides operations for sending an SMS message to the network, monitoring the delivery status of a sent SMS message, and asynchronously receiving notification of message delivery status. Parlay X SMS over Parlay interacts with a Parlay gateway using the Parlay Connector.

### Call flows

The following steps show an example of the call flow for Parlay X SMS over Parlay:

1. The application invokes a *sendSMS*.
2. Parlay X SMS over Parlay sends the *sendSMS* through the Parlay Connector to the Parlay gateway.
3. The Parlay gateway returns a *sendMessageRes* response through the Parlay Connector to Parlay X SMS over Parlay. In case of any error occuring during the processing of *sendSMS*, the gateway sends a *sendMessageErr* indicating that an error has occurred. If the confirm delivery option is opted, a *messageStatusReport* will be sent by the gateway.
4. Parlay X SMS over Parlay sends the Transaction ID as a response to the application.

5. Parlay X SMS over Parlay sends the *SMSnotification* response to the application.

### Interfaces

**Note:** These interfaces are provided as Web services–except for SmsNotification, which is used as a client only.

**SendSms**

**sendSms**
Allows you to send an SMS request and monitor the status of that request.

**getSmsDeliveryStatus**
Requests the status of a previous SMS delivery request identified by requestIdentifier. The information on the status is returned in deliveryStatus, which is an array of status related to the request identified by requestIdentifier. The status is identified by a couplet indicating a user address and the associated delivery status.

**sendSmsLogo**
A request to send a Logo.

**sendSmsRingTone**
A request to send a Ringtone.

**SmsNotification**

**notifySmsReception**
Requests the status of a previous SMS delivery request identified by requestIdentifier. The information on the status is returned in deliveryStatus, which is an array of status related to the request identified by requestIdentifier.

**notifySmsDeliveryReceipt**
When an SMS message is sent to multiple terminals, requests a notification from each terminal. The notification indicates one of three outcomes: delivery was successful, time expired before the message could be delivered, and notification is not supported for the terminal.

**ReceiveSms**

**getReceivedSms**
Retrieves all of the SMS messages received that fulfill the criteria identified by registrationIdentifier. The method returns only the list of SMS messages received since the last time the method was invoked. (Older messages are removed from the server.)

**SmsNotificationManager**

**startSmsNotification**
Enables an application to request notifications for short messages online.

**stopSmsNotification**
Enables an application to stop receiving notifications for short messages online.

## Integrating with Parlay X Call Handling over Parlay

The Parlay X Call Handling over Parlay Web service provides a mechanism for an application to designate how calls are to be handled for a specific number.

Parlay X Call Handling over Parlay provides a rule-based processing capability that applications can access through the following set of operations: setRules, getRules, setRulesForGroup, and clearRules.

The Parlay X Call Handling over Parlay Web service is designed to provide a simplistic method for applications, at they relate to call behavior. The interface functions on a request/response message from the Web client to a Web service. When actual calls are routed, the Web service validates whether the calls are permitted according to a configured address.

### Call flows

The following is an example call flow showing how the Parlay X Call Handling over Parlay interfaces are invoked. In this example the call flow begins when the application client invokes the Web service by routing the SOAP message over the HTTP to the Telecom Web Services Access Gateway. The ESB acts as a Web service between the client application and the Call Handling service. The following steps show how the Parlay X Call Handling over Parlay is invoked:

1. You attempt to make a call from a Parlay gateway.
2. CallHandlingNotificationManager receives a start/stop event from the Parlay Connector.
3. CallHandlingRuleProcessor evaluates the rules, and the CHRule executes the rules.
4. The call negotiation is complete, and you can now start your phone conversation.

### Interfaces

**setRules**
Sets the Call Handling rules for the call destination address. If a set of rules is already in place for the address, then this operation will replace the old rules with the set provided in the operation. The address may not specify a group. If a group is specified, a PolicyException will return.

**setRulesForGroup**
Sets the Call Handling rules for multiple call destinations. If a set of rules are already in place for any of the addresses, then this operation will replace the old rules with the set provided in the operation. The addresses may include groups with arbitrary prefix for group which will be resolved in the Telecom Web Services Access Gateway for members that use the *tel*:, *sip*:, and *sips*: URIs.

**getRules**
Retrieves the Call Handling rules for the call destination. The address may not specify a group. If a group is specified, a PolicyException will return.

**clearRules**
Clears the Call Handling rules associated with the specified address. If rules have not been set for an address, then the operation silently ignores the request and does not return a fault message. The addresses may include groups with members using the *tel*: and *sip*: URIs.

**Note:** Wildcards cannot be used to specify addresses.

## Integrating with Parlay X Call Notification over Parlay

Parlay X Call Notification over Parlay notifies Web clients of specific call events for a specific called party.

Parlay X Call Notification over Parlay defines interfaces for CallDirection, CallNotification, CallDirectionNotificationManager, and CallNotificationManager.

Parlay X Call Notification over Parlay is a Parlay X Web service application that sends Web service requests to the network mapper for the creating, registering, and reporting of network related events. Additionally, the Call Notification component processes information from the network elements that represent the event notification data, then passes that information on to the registered subscribers.

## Call flows

The following is an example call flow showing how the Parlay X Call Notification over Parlay interface is invoked. In this call flow example, the call flow begins when an incoming notification arrives. The following steps show how the CallNotification would be invoked:

1. A startCallNotification or startCallDirectionNotification request is processed.
2. The CNManagerCallBack notification is created.
3. The createNotification CNManagerCallback request is implemented.
4. The assignmentID is returned.
5. A database entry is created, containing the assignment ID and correlator.
6. The notificationInfo is returned.

## Interfaces

**CallDirection**

**handleBusy**
Sends a busy notification to the Web server when an event type is busy.

**handleNotReachable**
Sends a notification to the Web server when an event cannot be reached and the monitor mode is interrupted.

**handleNoAnswer**
Sends a notification to the Web server when an event is not answered.

**handleCalledNumber**
Sends a notification to the Web server when an event type is addressed and analyzed.

**CallNotification**

**notifyBusy**
Sends a notification to the Web server when an event is busy.

**notifyNotReachable**
Sends a notification to the Web server when an event type cannot be reached.

**notifyNoAnswerRequest**
Sends a notification to the Web server when the event type is not answered.

**notifyCalledNumber**
Sends a notification to the Web server when the event type is addressed and the monitor mode is set to notify.

**StartCallDirectionManager**

**startCallDirectionNotification**
Allows the availability of notification specific address network events.

**stopCallDirectionNotification**
Allows an application to end the notification for a specific address.

**StartCallNotificationManager**

**startCallNotification**
Delivers the notification call from the application to the receiving notification.

**stopCallNotification**
Stops the notification call from going to the application to the receiving notification.

## Integrating with Parlay X Third Party Call over Parlay

Parlay X Third Party Call over Parlay enables applications to send Web service requests to the Telecom Web Services Server requesting third party call services.

Parlay X Third Party Call over Parlay Using this Web service application developers can quickly develop applications use without having detailed knowledge of the telecommunications field. The Parlay X Third Party Call over Parlay provides Parlay X Web service support for applications with the ability to initiate a call between two addresses (makeCall), retrieves the current call status after you have initiated a call (getCallInformation), terminate a call (endCall), or cancel a call if the call is not connected (cancelCall).

### Call flows

This is a sample call flow for the makeCall method:

1. The application client invokes a flow that sets up a voice call between two addresses, callingParty and calledParty, provided that the invoking application is allowed to connect them.
2. After invoking the flow, the application monitors that status of the requested call. The callIdentifier parameter is used to identify the call.

This is a sample call flow for the getCallInformation method:

1. The application client retrieves the current status of the identifiable call.
2. After the call has ended, the status information is available for a limited amount of time. It is specified by the StatusRetainTime service policy.

This is a sample call flow for the endCall method:

1. The application client terminates the call identified by the callIdentifier.

This is a sample call flow for the cancelCall method:

1. The application cancels the previous call request identified by the callIdentifier.

### Interfaces

**ThirdPartyCall**

**makeCall**
Initiates a call between two addresses, CallingParty and CalledParty. Optionally the application can also indicate the charging information (Charging). This is determined by the policy information configured per provider.

**getCallInformation**
Retrieves the current call status of the CallIdentifier (a parameter returned from makeCall). A CallInformation structure is returned, containing StartTime, Call status, Duration, and TerminationCause (if applicable). Provided that the call status record has not expired and has not been purged, this method can be invoked multiple times even if the call has ended.

**endCall**
Terminates the call identified by CallIdentifier.

**cancelCall**
Cancels the call identified by CallIdentifier. This operation has no effect if a call has been connected.

Communication between nodes is facilitated when each endCall or cancelCall request is processed by the same node that initiated the corresponding makeCall request. To achieve this, endCall and cancelCall use `ClusterCommn.jar`—a common utility file that is provided when you install the Parlay X Third Party Call over Parlay Web service implementation. Its function is to forward each endCall or cancelCall request to the node that initiated the makeCall.

## Integrating your applications with the Access Gateway

If you use a Web services description language (WSDL) to develop applications that integrate with the Telecom Web Services Server, the WSDL must meet certain requirements.

All communication between your application and the Access Gateway is done using SOAP/HTTPS. This provides maximum interoperability between the endpoints, because literal encoding has the widest platform support and interoperability characteristics. Communication between the Access Gateway and the back-end service implementation uses SOAP over the most appropriate transport. Both HTTP and secure HTTPS transports can be used for delivering the SOAP message.

The WSDL you use to develop an application for integrating with the Telecom Web Services Server must meet the following requirements:
- The WSDL must have the encoding type of *Document/Literal wrapped*.
- The elements name and type are required.
- The name and type attributes should be the same, for example StartCallNotification.
- The ComplexType must have the same value and element name.

## Access Gateway message processing flows

Telecom Web Services Server (TWSS) includes a set of default flows. In many cases you will be able to use the default flows that are provided. However, it is possible to create your own custom flows or modify the default flows by adding new mediation primitives.

### Using the Telecom Web Services Server plug-in for WebSphere Integration Developer

The WebSphere Integration Developer (WID) environment is used to customize message processing flows associated with the Access Gateway.

Installing the Telecom Web Services Server (TWSS) plug-in for WID is optional for TWSS installations. It is required, however, if you intend to customize the Access Gateway mediation flows.

## Installing the Telecom Web Services Server plug-in for WebSphere Integration Developer

Follow these steps to download and install the Telecom Web Services Server (TWSS) plug-in for WebSphere Integration Developer (WID).

### Before you begin

You must have completed the following steps:

- Obtained and installed WebSphere Integration Developer, Version 6.1.0.102
- Updated the WID tooling to Version 6.1.0.102 or later

### About this task

The TWSS plug-in for WID includes the following components:

- Access Gateway default flow mediation project interchange files that can be imported into WID for customization
- TWSS mediation primitive plug-ins that extend the WID mediation flow editor palette with custom mediation primitives that have been included in the Access Gateway default flows
- Post-processing scripts that must be run against customized Access Gateway default flow EAR files that are exported from WID (the scripts re-add deployment code that is overwritten by WID tooling when the mediation project interchange file is imported into WID)

In the TWSS WID installer, the WID plug-in is on a separate CD from the TWSS base installer and the TWSS services installer. The following directory structure is on the CD:

```
/widplugin
  /Linux
    setup
    twss_wid_install.rsp
    twss_wid_uninstall.rsp
  /Linuxppc
    setup
    twss_wid_install.rsp
    twss_wid_uninstall.rsp
  /Win32
    setup.exe
    twss_wid_install.rsp
    twss_wid_uninstall.rsp
```

To download and install the Telecom Web Services Server plug-in for WID, follow these steps.

1. Stop the WID servers, such as the WebSphere ESB server.
2. Shut down WID.

   The TWSS plug-in installer for WID cannot programmatically shut down the WID workbench. Therefore, you must close and exit WID manually.

   If the plug-in installer continues to display warnings that WID is still started after you have shut it down, additional IBM Rational products may be sharing common executable code with WID.

The IBM Rational Software Development Platform (RSDP) is a common development environment that is shared by several products, including WID, Rational Application Developer, Rational Software Architect, and others. If you install any of these products, the RSDP is installed automatically, but it is installed only once. As a result, if WID was installed first, it shares its development platform with other Rational products.

Stop any other Rational products before continuing with the installation of the Telecom Web Services Server plug-in for WID.

3. Download the Telecom Web Services Server plug-in for WID, `TWSS_WIDPlugin_C14CAEN.zip`, to a temporary directory. The WID plug-ins installer is a separate CD image that is part of the Telecom Web Services Server product offering.
4. Extract (unzip) the contents of the file.
5. Open the `/WIDPlugin` directory and navigate to the directory for the operating system where WID is installed–for example, `/Linux` or `/Win32`.
6. Launch the `setup` routine to begin the installation.
7. The Welcome panel displays with English as the default. Click **OK**.
8. After the installer initializes, click **Next** through the License Agreement and Overview windows.
9. When you are prompted for the WebSphere Server home directory, browse to the *`WID_home`* directory. Then click **Next**.
10. When you are prompted to select which features to install, check the box for **TWSS WID plug-in**. Then click **Next**.
11. Verify the installation path and the features you have selected. When you are satisfied with the selections, click **Next**.
12. When the installation has completed, click **Finish**.

### Results

To verify that the installation completed successfully, refer to the topic *Verifying the installation of the WebSphere Integration Developer plug-in*.

**Note:** Refer to the topic *Troubleshooting WID* for details about resolving problems with installing the WID plug-in.

## Installing the Telecom Web Services Server plug-in for WebSphere Integration Developer (WID) silently

The IBM WebSphere Telecom Web Services Server (TWSS) plug-in for WID can be installed silently without user interaction, by reading user responses from a response file.

### About this task

A silent installation uses the TWSS plug-in installer for WID to install the product in silent mode, without the graphical user interface. Instead of displaying a wizard interface, the silent installation causes the installation program to read all of your responses from a response file that you provide.

To specify non-default options during a silent installation, customize the response file. To install silently, you must accept the license agreement using the agreement option within the response file.

**Note:** Refer to the topic *Troubleshooting WID* for details about resolving WID plugin installation issues.

**Customizing the installation response file for the plug-in:**

Before invoking the Telecom Web Services Server (TWSS) plug-in installer for WID in silent mode, you need to customize the response file that will be used to provide user responses to the installer.

**About this task**

Use the response file to supply values to the TWSS plug-in installer for WID. Running in silent mode, the plug-in installer does not display interactive dialogs. Instead, it reads values from the response file.

Be precise when supplying values in the file: Incorrect specifications affect the silent interface of the plug-in installer.

Follow these steps to customize the response file for the TWSS plug-in installer silently:

1. Locate the response file *`twss_wid_install.rsp`* on the installation CD: The sample response file is in the *`response_files`* directory.
2. Copy the response file to the machine where you intend to install the WID plug-in.
3. Open the sample response file in a text editor of your choice.
4. Edit the response file properties.

*Table 38. The following table lists the response file parameters.*

| Property | Description | Example values |
|---|---|---|
| LICENSE_ACCEPTED | The license must be accepted before the installation process. | true |
| WAS_HOME | The target installation home directory of the WebSphere server. | /opt/IBM/WID |

   **Note:** The response file parameters have to be configured.

5. Save the edited document.

**Results**

You are now ready to run the plug-in installer, which will use the edited response file to get the settings it needs.

**What to do next**

When the silent installation has completed, check the appropriate logs in the *`WID_home/logs`* directory (where *WID_home* is the WID home directory). Refer to the topic *Monitoring log messages* for additional information about reading logs.

You are now ready to run the installation script. The script will use the edited response file to get the settings it needs.

**Installing the plug-in silently:**

When you install the Telecom Web Services Server (TWSS) plug-in for WebSphere Integration Developer (WID), a silent installation can be run by executing the setup script using a customized response file.

**Before you begin**

Configure the installation by changing the values of the parameters in the response file. Refer to the topic *Customizing the installation response file for the plug-in*.

**About this task**

Follow these steps to install the TWSS plug-in for WebSphere Integration Developer silently:

1. Log on to the operating system with a user ID that has administrator authority, such as **root**.
2. Select a umask that allows the owner to read and write to the files, and that allows others to access them according to the prevailing system policy.

   **Note:** For root, a umask of **022** is recommended. For non-root users an umask of **002** or **022** can be used, depending on whether or not the users share the group.
3. Invoke the setup script to install the Telecom Web Services Server plug-in for WID. The script makes use of your customized response file. Issue the following command:

   ```
   ./setup -silent-options full_path_to_response_file
   ```

   For example:

   ```
   ./setup -i silent -f /opt/IBM/twss_wid_install.rsp
   ```

**What to do next**

When the silent installation has completed, check the appropriate logs in the `WID_home/logs` directory (where *WID_home* is the WID home directory). Refer to the topic *Monitoring log messages* for additional information about reading logs.

You are now ready to run the installation script. The script will use the edited response file to get the settings it needs.

## Verifying the installation of the Telecom Web Services Server plug-in for WebSphere Integration Developer

You can verify the installation of the plug-in by verifying that the correct directories and files have been added to the `WID_home` directory.

### About this task

To verify that the Telecom Web Services Server (TWSS) Access Gateway default flow project interchange files and post-processing scripts were installed, follow these steps.

1. Navigate to the `WID_home` directory (the directory in which WID is installed).

   If the installation was successful, you will see the following subdirectory:
   - `uninst_TWSS-WID`
2. Navigate to the `WID_home/TWSS` directory. The directory should contain the following files:

   ```
   PX21_AC_FLOW.zip
   ```

```
PX21_ALM_FLOW.zip
PX21_AM_FLOW.zip
PX21_CH_FLOW.zip
PX21_CN_FLOW.zip
PX21_MC_FLOW.zip
PX21_MM_FLOW.zip
PX21_PMT_ACS_FLOW.zip
PX21_PMT_RVCS_FLOW.zip
PX21_PRS_FLOW.zip
PX21_SMS_FLOW.zip
PX21_TL_FLOW.zip
PX21_TPC_FLOW.zip
PX21_TS_FLOW.zip
WAP10_FLOW.zip
flowearpostproc.bat
flowearpostproc.xml
esb_set_app_role_xsl
esb_set_flow_rootctx_xsl
esb_set_flow_urlmap_xsl
esb_set_servlet_role_xsl
twss_insert_handler.xsl
```

The .bat exists instead of a shell script for Linux and the batch file for Windows®.

3. To verify that the Telecom Web Services Server mediation primitive plug-ins were installed, navigate to the *`WID_home`*`/plugins` directory. The directory should contain the following subdirectories:

```
com.ibm.twss.ag.mediation.plugin.addressmasking_7.0.0.0
com.ibm.twss.ag.mediation.plugin.groupresolution_7.0.0.0
com.ibm.twss.ag.mediation.plugin.jmxnotification_7.0.0.0
com.ibm.twss.ag.mediation.plugin.messageinterceptor_7.0.0.0
com.ibm.twss.ag.mediation.plugin.msgelementremover_7.0.0.0
com.ibm.twss.ag.mediation.plugin.networkstatistics_7.0.0.0
com.ibm.twss.ag.mediation.plugin.policyretrieval_7.0.0.0
com.ibm.twss.ag.mediation.plugin.serviceauthorization_7.0.0.0
com.ibm.twss.ag.mediation.plugin.serviceinvocation_7.0.0.0
com.ibm.twss.ag.mediation.plugin.slaclusterenforcement_7.0.0.0
com.ibm.twss.ag.mediation.plugin.slalocalenforcement_7.0.0.0
com.ibm.twss.ag.mediation.plugin.transactionidentifier_7.0.0.0
com.ibm.twss.ag.mediation.plugin.transactionrecorder_7.0.0.0
```

## Results

A record file can be generated after the completion of an installation or uninstallation. The file records the values and options that were selected during the interactive installation and can be used for subsequent silent installations.

To generate an install record file, issue the following command after the interactive installation completes:

```
./setup -options-record recordFileName
```

To generate an uninstall record file, issue the following command after the interactive uninstallation completes:

```
./uninstall -options -record recordFileName
```

### What to do next

Note that the Telecom Web Services Server mediation primitive plug-ins are not added to the WID mediation flow editor palette until WID tooling is started using the `-clean` option, for example:

Linux

```
./WID61/wid.sh -clean
```

```
C:\WID61\wid.exe -clean
```

## Developing and customizing a custom Access Gateway flow

The WebSphere Integration Developer (WID) environment is used to customize the Parlay X message processing flows. Customizing the flows is optional and depends on the needs of your environment. WID tooling is also necessary to create custom flows for non-Parlay X Web services.

### Modifying a default mediation flow

A default mediation primitive flow is modified by adding a custom mediation primitive or by removing any of the Telecom Web Services Server (TWSS) mediation primitives from the flow.

#### Before you begin

You will use one of the flows that is provided as part of your TWSS plug-in for WebSphere Integration Developer. The Parlay X PresenceSupplier interface is used here to modify the flow.

1. Import the **PX21_PRS_FLOW.zip** project interchange file.
2. Add a custom primitive to the flow and implement a sample business logic.
3. Export the flow and then deploy it to the runtime environment.

   **Note:** For development of new mediation primitives, refer to the topic *Creating custom mediation primitives*.

### Creating custom mediation primitives

You can use WebSphere Integration Developer (WID) to create custom mediation primitives using Java snippets.

Mediation primitives implement a pipelined architecture, where information can be passed between primitives in the form of SOAP headers. Here are some guidelines for designing and implementing custom mediation primitives to best fit into this architecture.

- Mediation primitives should contain narrowly defined, well scoped function. This model encourages different mediation primitives for different variations of

function, rather than complex configuration. The best primitive can be chosen (or substituted) to meet a specific need when you are constructing a flow.
- Try to minimize the upstream assumptions of mediation primitives. By minimizing assumptions about headers from elements that are higher in the pipeline, you gain flexibility of use.
- Mediation primitives should be as policy driven as possible. Policy information represents runtime configuration and can change constantly. This allows for centralized administration of services through the service policy management system.

In the WID environment, a “Custom Mediation Primitive” is placed onto the canvas and you use a code editor to insert Java mediation code. For details about this process, refer to the topic *Implementing custom mediation logic* in the WebSphere Integration Developer information center.

Although this methodology provides a quick and easy way to create a custom mediation primitive, it has limitations for reuse. Each project must import the custom mediation primitive call and recreate the addition of the custom primitive.

To allow greater reuse of your custom mediation primitives, you can package them in such a way as to extend the WID Flow Editor palette. Considerations for doing this are described in the topic *Integrating your custom mediation primitives with WebSphere Integration Developer (WID)*.

## Creating Access Gateway mediation subflows

A mediation subflow is a preconfigured set of mediation primitives that are sequenced together to represent a series of steps in a mediation flow. With subflows, you can more easily customize the default Access Gateway flows and apply your customizations across multiple flows.

A subflow is analogous to a snippet or a macro that is used within a larger routine. When you insert a subflow into your Access Gateway flows, you can change all of the flows simply by changing the one subflow–rather than by having to change each individual flow. The change in the subflow is propagated to every flow in which the subflow is embedded.

Several subflows are provided for your use with the Telecom Web Services Server version 7.0 product.

#### RequestCommonFlow:

RequestCommonFlow is used for Web services that do not support groups. It resembles the default Access Gateway request message processing flow, except that it does not include the Group Resolution mediation primitive.

The RequestCommonFlow subflow executes mediation primitives in the following sequence:

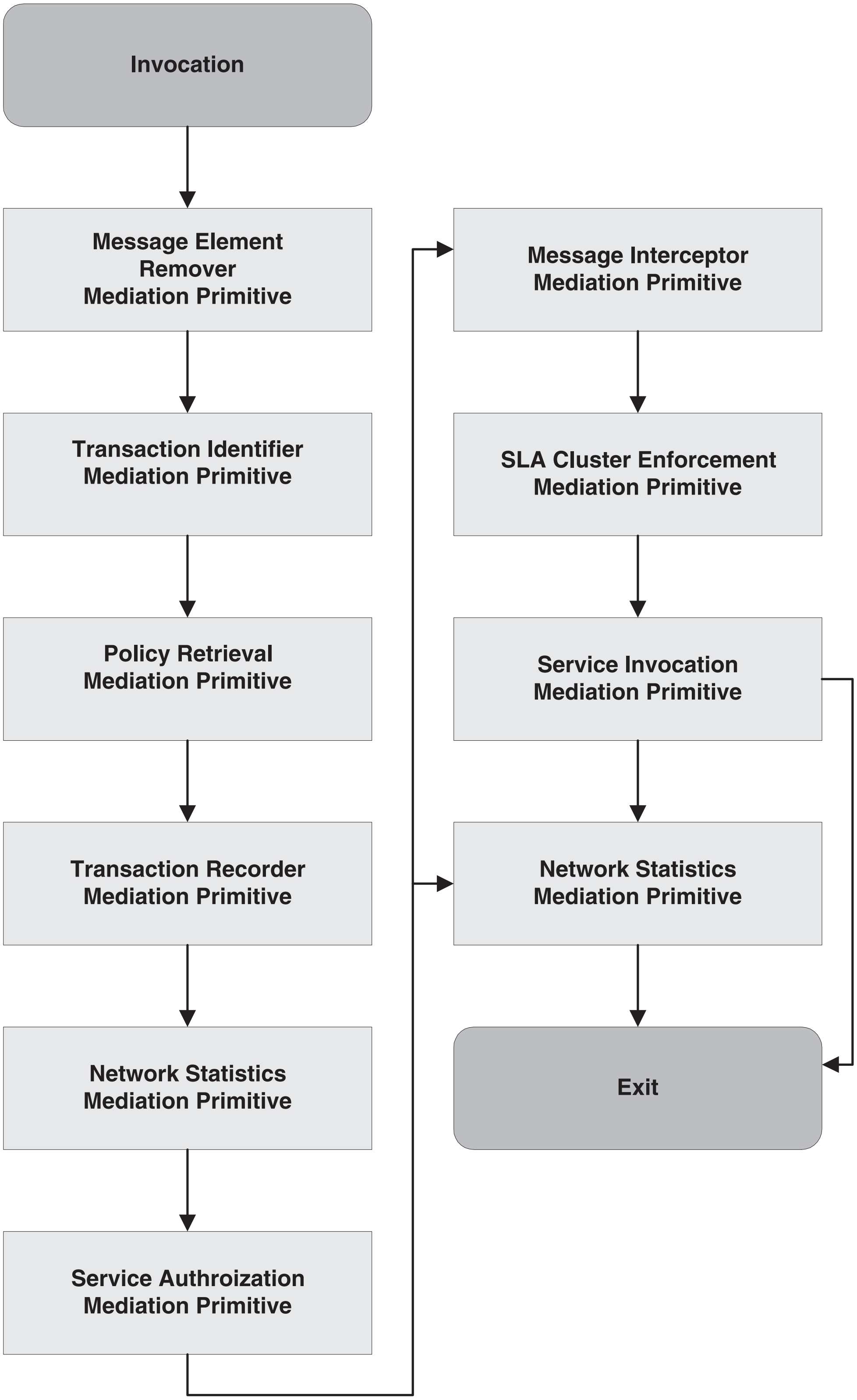

Invocation
Message Element Remover Mediation Primitive
Transaction Identifier Mediation Primitive
Policy Retrieval Mediation Primitive
Transaction Recorder Mediation Primitive
Network Statistics Mediation Primitive
Service Authroization Mediation Primitive
Message Interceptor Mediation Primitive
SLA Cluster Enforcement Mediation Primitive
Service Invocation Mediation Primitive
Network Statistics Mediation Primitive
Exit

*Figure 1. RequestCommonFlow subflow*

As the figure shows, a typical RequestCommonFlow subflow includes an input node (in) followed by a sequence of TWSS mediation primitives, an output node (out), and a failure node (out_Fail). In TWSS default flows the output node is connected to the Callout node of the main flow and the failure node is connected to the RequestFailureFlow subflow. (Refer to the topic *RequestFailureFlow* for a complete description of that subflow.)

**RequestCommonFlowWithGR:**

RequestCommonFlowWithGR is used for Web services that support groups. It resembles the default Access Gateway request message processing flow.

The RequestCommonFlowWithGR subflow is very similar to RequestCommonFlow, the only difference being that RequestCommonFlowWithGR includes the Group Resolution mediation primitive.

The RequestCommonFlowWithGR subflow can be used for all Web services that support groups. Even when the subflow is used in this way, you have the option to bypass the Group Resolution mediation primitive by specifying the null value in the **Group Resolution Endpoint** promoted property.

The RequestCommonFlowWithGR subflow executes mediation primitives in the following sequence:

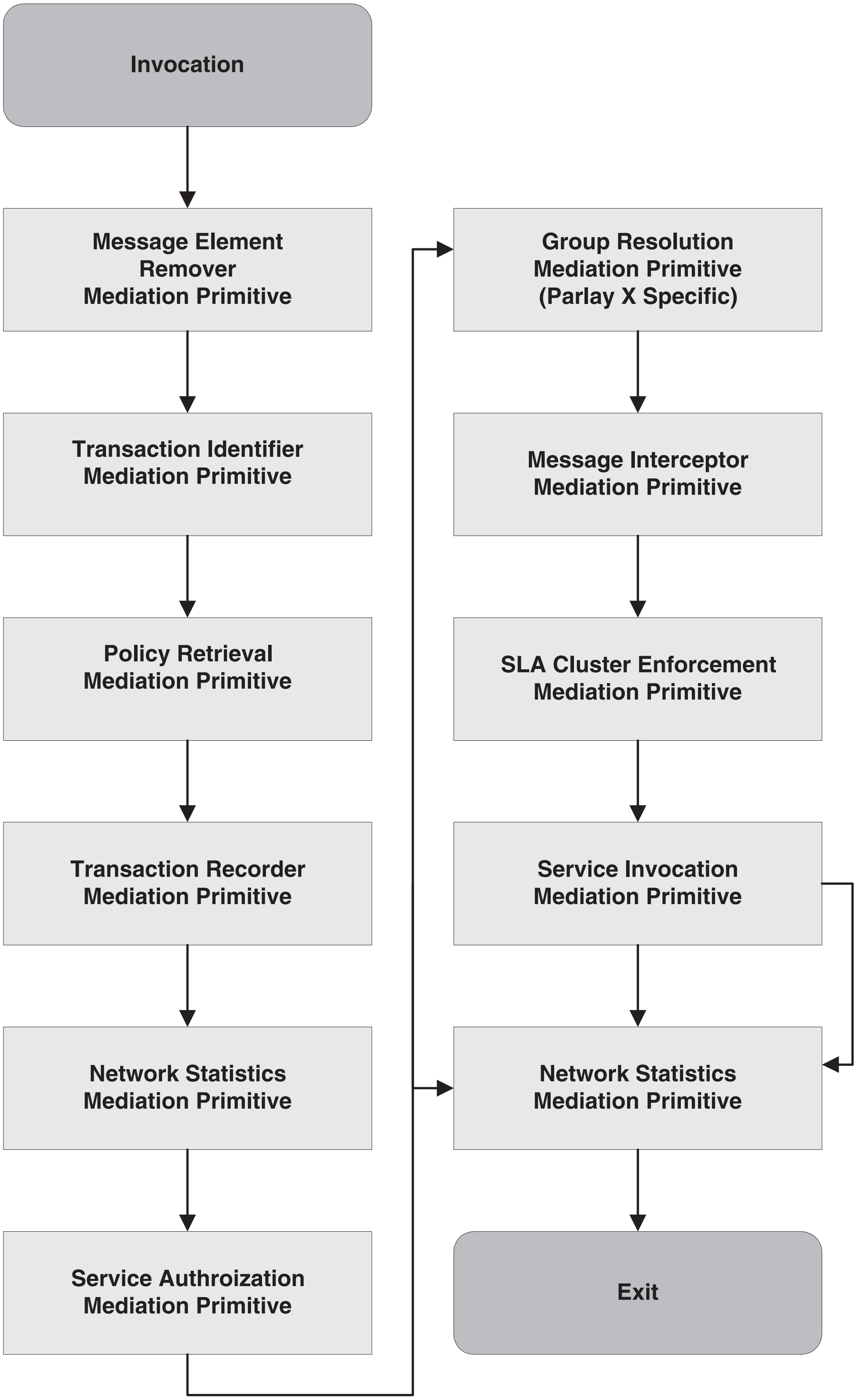

Invocation
Message Element Remover Mediation Primitive
Transaction Identifier Mediation Primitive
Policy Retrieval Mediation Primitive
Transaction Recorder Mediation Primitive
Network Statistics Mediation Primitive
Service Authroization Mediation Primitive
Group Resolution Mediation Primitive (Parlay X Specific)
Message Interceptor Mediation Primitive
SLA Cluster Enforcement Mediation Primitive
Service Invocation Mediation Primitive
Network Statistics Mediation Primitive
Exit

*Figure 2. RequestCommonFlowWithGR subflow*

As the figure shows, a typical RequestCommonFlowWithGR subflow includes an input node (in) followed by a sequence of TWSS mediation primitives, an output node (out), and a failure node (out_Fail). In TWSS default flows the output node is connected to the Callout node of the main flow and the failure node is connected to the RequestFailureFlow subflow. (Refer to the topic *RequestFailureFlow* for a complete description of that subflow.)

**RequestFailureFlow:**

RequestFailureFlow is used as the failure flow in the default request flow of Access Gateway mediations.

The RequestFailureFlow subflow executes mediation primitives in the following sequence:

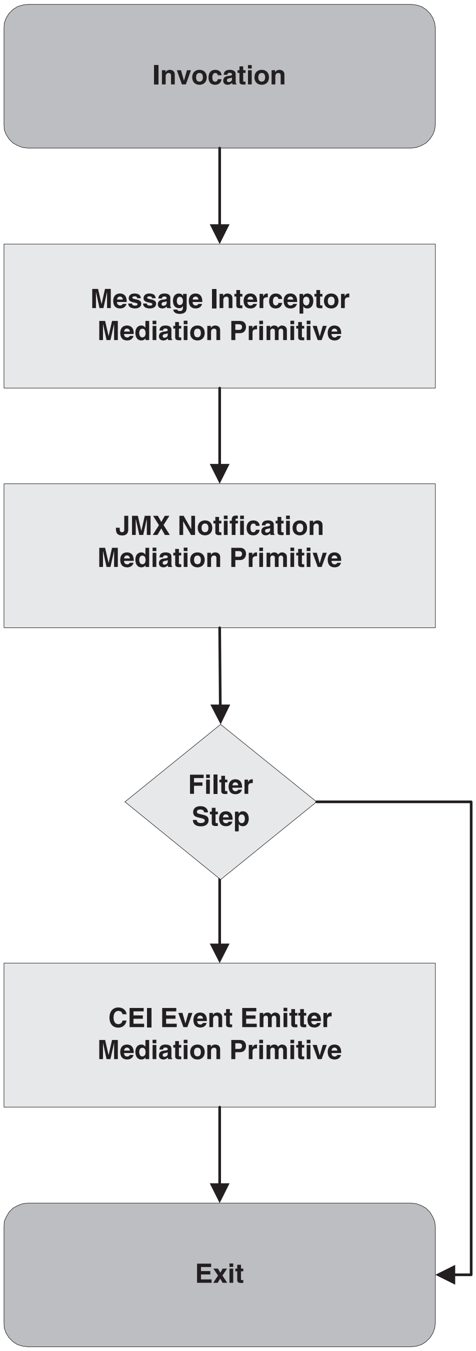

Invocation
Message Interceptor Mediation Primitive
JMX Notification Mediation Primitive
Filter Step
CEI Event Emitter Mediation Primitive
Exit

*Figure 3. RequestFailureFlow subflow*

As the figure shows, a typical RequestFailureFlow subflow includes an input node (in) followed by a sequence of TWSS mediation primitives and an output node (out). In TWSS default flows, if a failure occurs during the execution of a RequestCommonFlow or RequestCommonFlowWithGR subflow, then RequestFailureFlow is invoked to handle the fault condition.

**ResponseFlow:**

ResponseFlow is used as the response flow in the default flow of Access Gateway mediations.

The ResponseFlow subflow executes mediation primitives in the following sequence:

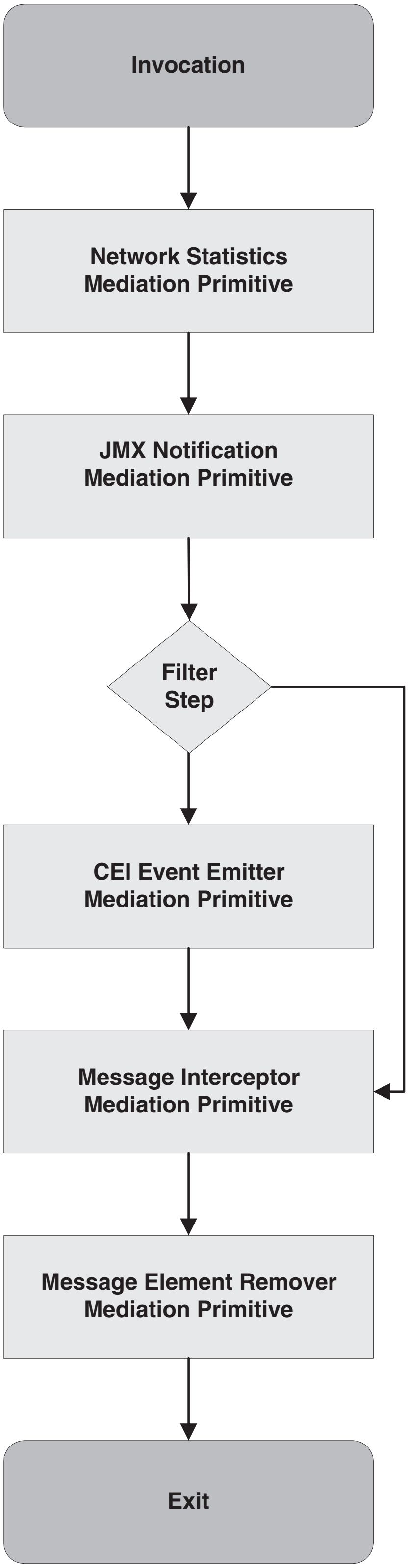

Invocation
Network Statistics Mediation Primitive
JMX Notification Mediation Primitive
Filter Step
CEI Event Emitter Mediation Primitive
Message Interceptor Mediation Primitive
Message Element Remover Mediation Primitive
Exit

*Figure 4. ResponseFlow subflow*

As the figure shows, a typical ResponseFlow subflow includes an input node (in) followed by a sequence of TWSS mediation primitives and an output node (out). When the response is received from the partner node, the ResponseFlow subflow is executed. This ResponseFlow subflow is responsible for handling different kinds of responses including a success, a failure, or defined exceptions

**Examples of Access Gateway mediation subflows:**

The following example shows mediation subflows being used in Access Gateway request and response flows.

**Example: message processing for sendSms operation using subflow**

On receipt of a sendSms message, the RequestCommonFlow subflow is executed. If message processing is successful the request is sent to the callout, which typically calls the Web service implementation.

If message processing is not successful, the message is routed trough the fail terminal of the RequestCommonFlow subflow to RequestFailureFlow. Then the message is inspected to find the type of exception that occurred. An XSLT mediation primitive converts the message to a PolicyException or ServiceException, and the message is returned to the caller through the InputFault terminal.

The Web service implementation can respond to the sendSms request message in one of three ways: success, failure, or exception.

- If the Web service implementation returns a successful response, the response message received is routed through the ResponseFlow subflow back to the caller.
- If the Web service implementation fails to respond, the an XSLT mediation primitive converts the request message to a ServiceException, which is then routed through the ResponseFlow subflow back to the caller.
- If the Web service implementation returns a Service or Policy exception, the exception message received from the respective Callout Fault terminals is routed through the ResponseFlow subflow back to the caller as a Service or Policy exception.

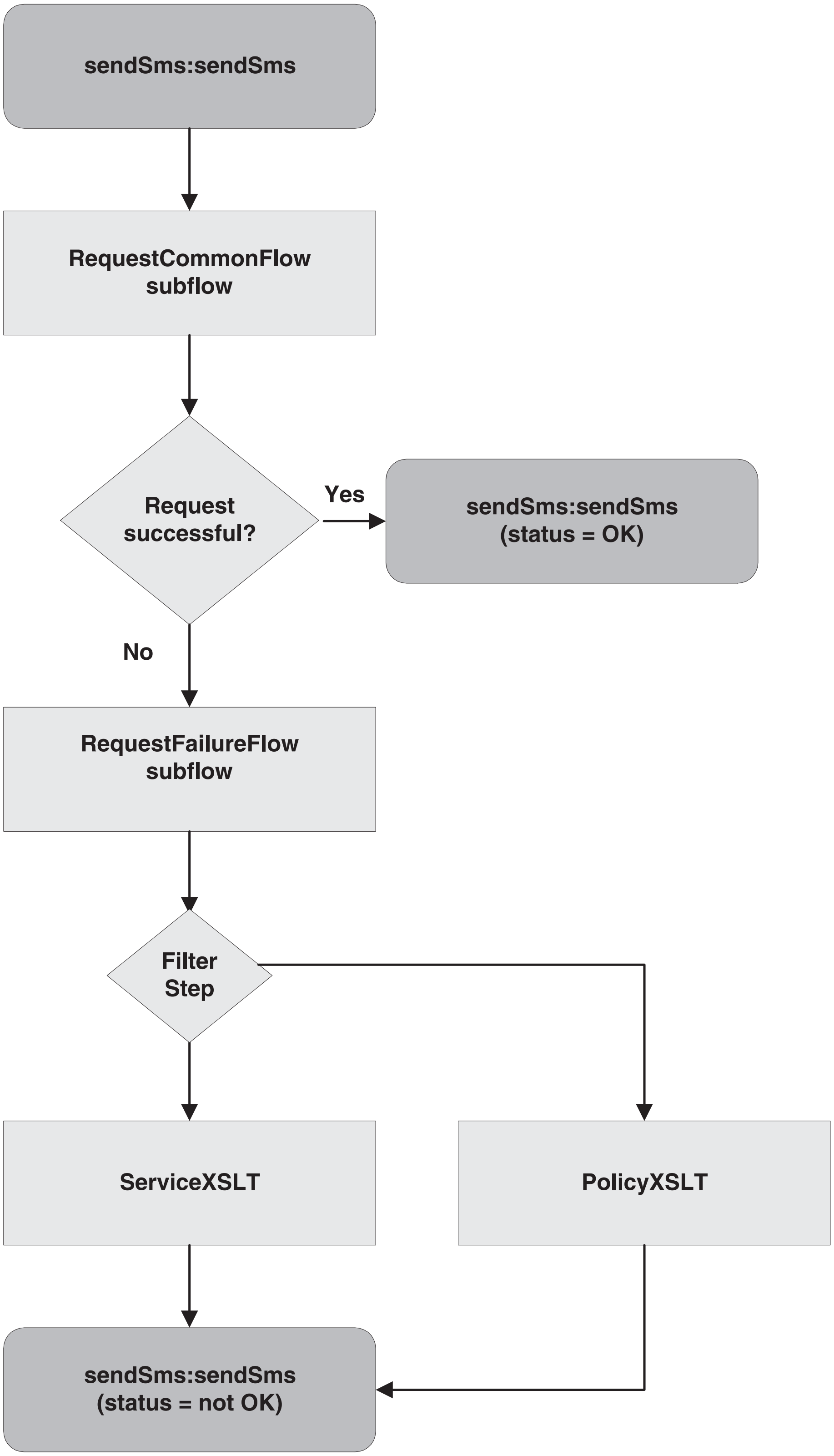

sendSms:sendSms
RequestCommonFlow subflow
Request successful?
Yes
sendSms:sendSms (status = OK)
No
RequestFailureFlow subflow
Filter Step
ServiceXSLT
PolicyXSLT
sendSms:sendSms (status = not OK)

*Figure 5. Example of a request flow using mediation subflows*

### Integrating your custom mediation primitives with WebSphere Integration Developer (WID)

A reusable approach for creating custom mediation primitives is to package the mediation primitives so that they plug in to the Flow Editor palette. The palette is part of the Telecom Web Services Server plug-in for WebSphere Integration Developer (WID).

To integrate your custom mediation primitives into the Flow Editor palette, you must create a JAR file that contains the mediation primitive runtime code. Then you must place that code within the class path for WebSphere Enterprise Service Bus.

Eclipse plug-in code is also needed, and an icon must be included for the Flow Editor palette. After they are installed into the WID Eclipse plug-in directory, these plug-ins are automatically populated into the palette when WID is started with the **-clean** option.

Mediation primitives that extend the Flow Editor palette can also have static properties that can be set. These properties are described within the Eclipse plug-in descriptor and appear within the Properties view below the Assembly Diagram canvas. Properties can be set on each primitive instance that is added into the canvas.

Property values can be set only within the Flow Editor and are stored within the EAR file that is generated from the flow artifacts. Because of this, custom mediation primitives should use properties only for static information, such as JNDI resource names or default properties. All other configuration information should be policy driven.

For more information about developing custom mediation primitives, refer to the topic *Implementing custom mediation logic* in the WebSphere Integration Developer information center.

## Reference for Access Gateway message processing flows

The default message processing flows reflect a typical processing function. Each default flow invokes the mediation primitives provided with the Access Gateway in a single sequence or in a set of sequences.

Each default flow is designed to support the accounting of requests, service or operation level authorization, message capture for regulatory purposes, and traffic level enforcement.

Should you decide to create a custom flow or modify a default flow, Telecom Web Services Server (TWSS) provides support for WID tooling and SOAP attachments.

### Mediation primitives

Mediation primitives plug in to the WebSphere Integration Developer (WID tooling palette. They can be used as primitive operations when you are creating Access Gateway mediation flows.

#### Introduction

A set of default flows is provided when you install the Access Gateway. To create a customized logic, you can utilize the WID tooling to assemble flows among the different components.

Each mediation primitives consists of Java code that implements a component interface for the Access Gateway. A mediation primitive is a logical unit that has a failure terminal and a number of input and output terminals.

Data objects are passed during execution of an Access Gateway flow between I/O terminals. The data objects use a representation called Service Message Object (SMO), for the SOAP (Simple Object Access Protocol) message that is being processed for a Web service invocation. The SMO object arrives at a mediation primitive's input terminal during execution flow, which allows control to be passed on to the primitive's mediate method.

The mediation primitive can then process the message and determine where to place the response on its defined output terminal, according to the semantics of that terminal. Mediation flows are created by wiring or connecting the input and output terminals.

Mediation primitives are the primary programming model for the Access Gateway subcomponents.

Mediation primitives use a pipeline architecture. You can insert additional mediation primitives anywhere in the flow, change the order of execution (subject to some constraints), and so on. The pipeline architecture is based on the SOAP processing model, where additional SOAP headers may be added, modified, or removed prior to passing execution to the next downstream element.

Each mediation primitive has unique semantics or expectations to define the input at its terminal and to specify the data that must be present in order to execute its logic. For example, a mediation primitive might insert additional headers to a SOAP message for processing by downstream elements, or it might require upstream primitives to insert headers. These semantics or expectations can result in constraints on how a particular component is used within a mediation flow.

## Mediation primitives used by the Access Gateway

The following components are considered mandatory for a base Access Gateway configuration and flow:

- Transaction Identifier mediation primitive
- Policy Retrieval mediation primitive
- Service Invocation mediation primitive

The mandatory components provide base services that support mediation primitives. The Transaction Identifier mediation primitive records information about the transaction within a table that is typically referenced by the Network Statistics mediation primitive. The Policy Retrieval mediation primitive retrieves policy data based on the requester, service, and operation being called. This data is used as decision parameters within the mediation primitive's execution flow.

The following components are optional plug-ins and are used by the default Access Gateway flow:

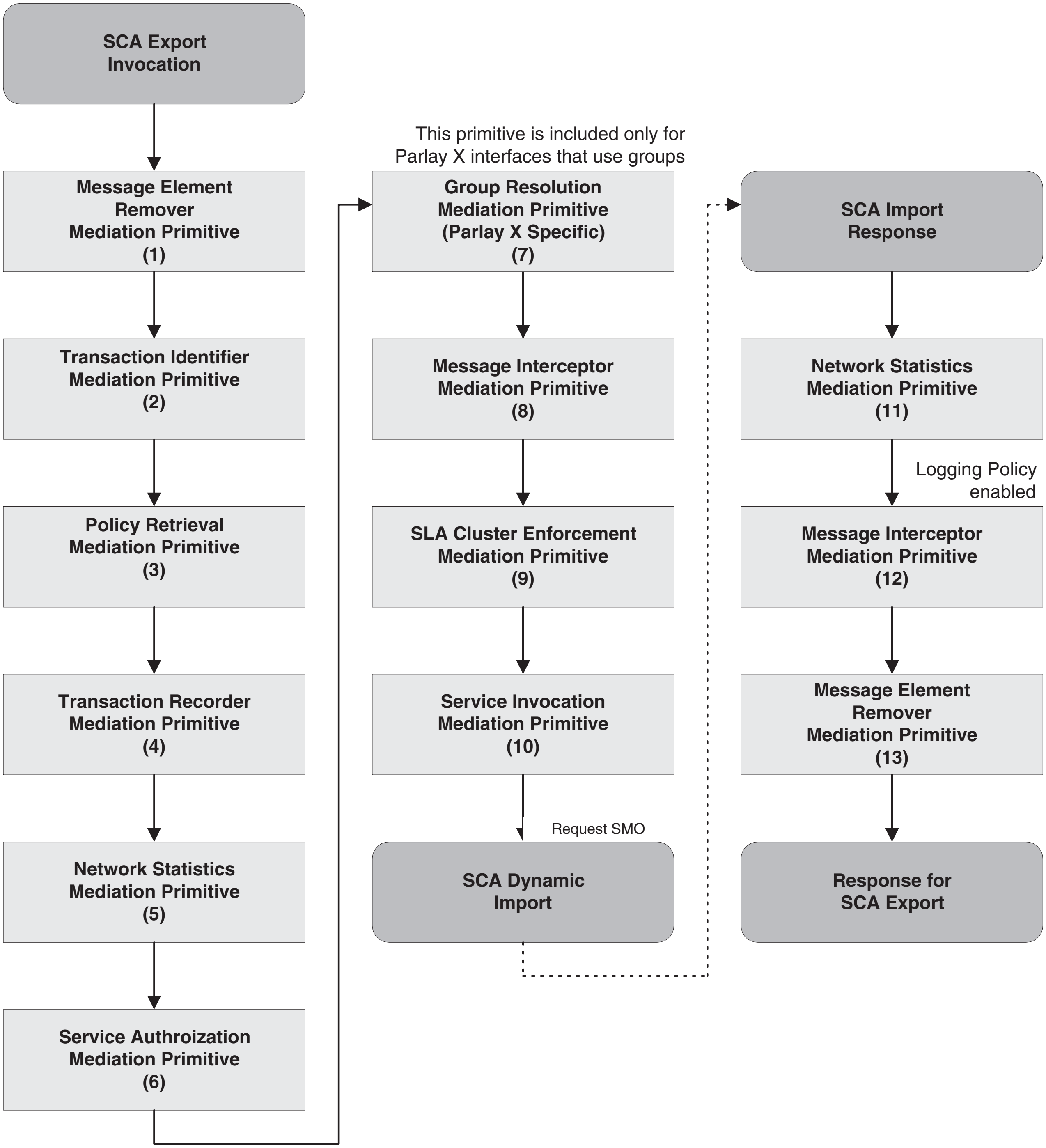
- Message Element Remover mediation primitive
- Address Masking mediation primitive
- Transaction Recorder mediation primitive
- Network Statistics mediation primitive
- Message Interceptor mediation primitive

- Service Authorization mediation primitive
- Group Resolution mediation primitive (Parlay X-specific)
- SLA Enforcement mediation primitives
- JMX Notification mediation primitive
- CEI Event Emitter mediation primitive (part of the WebSphere ESB product offering)

## Default Access Gateway flow

The default message processing flow for the Access Gateway makes use of the various mediation primitives in a single sequence or in a set of sequences.

Upon receipt of a Web service request, the following mediation primitives are executed in this sequence:


SCA Export Invocation
This primitive is included only for Parlay X interfaces that use groups
Message Element Remover Mediation Primitive (1)
Transaction Identifier Mediation Primitive (2)
Policy Retrieval Mediation Primitive (3)
Transaction Recorder Mediation Primitive (4)
Network Statistics Mediation Primitive (5)
Service Authroization Mediation Primitive (6)
Group Resolution Mediation Primitive (Parlay X Specific) (7)
Message Interceptor Mediation Primitive (8)
SLA Cluster Enforcement Mediation Primitive (9)
Service Invocation Mediation Primitive (10)
Request SMO
SCA Dynamic Import
SCA Import Response
Network Statistics Mediation Primitive (11)
Logging Policy enabled
Message Interceptor Mediation Primitive (12)
Message Element Remover Mediation Primitive (13)
Response for SCA Export

**Note:** To avoid a warning message when utilizing WebSphere Integration Developer (WID), the mediation flow format has been altered. This has been done to reduce the size and also improve the performance of the flow execution.

*Figure 6. Default Access Gateway Mediation Flow*

1. The Message Element Remover mediation primitive checks the user principal against the requester exception list. If the requester is not on the list, and if headers exist, the message element remover removes the TWSS headers from the request (using the appropriate XPath selector element).

2. The Transaction Identifier mediation primitive adds SOAP headers with the global transaction ID requester and name. If the global transaction ID and requester name have already been provided from an upstream element, then those values are used.
3. The Policy Retrieval mediation primitive retrieves the policy information from the Service Policy Manager.
4. If transaction recording is enabled by policy attributes, then the information about the transaction–such as the unique ID, requester, service, and operation–is recorded to a database table. (If the transaction recorder is disabled, then network statistics must also be disabled because of the database structure.)
5. If the Network Statistics mediation primitive is enabled, it records statistical information about the request. The Transaction Recorder mediation primitive must be enabled so that Network Statistics can record the details. The Network Statistics record corresponds to the Transaction Recorder's transaction ID.
6. If applicable, authorization is verified based on requester, service requested, and operation.
7. The request is checked to verify that it is a Parlay X request whose parameters may contain a list of URIs, some of which may be groups.
   - If it is not such a Parlay X request, skip to step 9.
   - If it is such a Parlay X request, URI groups expand into a flat list of member URIs, by resolving groups through the IBM WebSphere XML Document Management Server Component (IBM XDMS).
8. If message capture is enabled by policy, the Message Interceptor mediation primitive logs the required part of the message to a database. If address tracking is enabled by policy, the mediation primitive compares the addresses specified by the policy with the target addresses in the message. If a match is found, the target addresses are logged to the database with the corresponding message.
9. If the SLA Enforcement mediation primitive is enabled, then monitoring information is adjusted and the statistics information is checked to ensure that a request can be processed.
10. The backend Web service is invoked.
11. If enabled for the particular service operation, response statistics are recorded.
12. The filter step checks whether message capture is enabled by policy. If so, the message is logged to a database using the mediation primitive provided by WebSphere ESB.
13. The Message Element Remover mediation primitive checks the user principal against the requester exception list. If the requester is not in the list, and if TWSS headers exist, the Message Element Remover removes the TWSS headers element, using an appropriate XPath selector.

Upon completion of the flow, the response object is returned through the SCA export invocation, as the Web service response. If a failure occurs during the execution of the request portion of the flow, then fault handling segment of the flow is invoked. See the topic *Access Gateway fault flow* for details.

## Access Gateway fault flow

When errors are detected during the default message processing flow, the Access Gateway invokes certain mediation primitives in a set sequence.

Upon completion of the default message flow, the response object is returned through the SCA export invocation, as the Web service response. If a failure occurs during the execution of the request portion of the flow, then fault handling segment of the flow is invoked.

When handling a Web service response, the response is always returned to the requester despite faults within the flow. This ensures consistency with the backend system. Errors during a request flow are signified by the output of the mediation primitive being placed on its fault terminal. This fault terminal will be wired to the fault processing segment of the flow, which returns the error response to the requester.

The Access Gateway performs a transformation step, in order to convert a fault to a WSDL defined error type in a fault situation. This step is performed just prior to returning the error response to the Web service client, and it matches the client expected output from the WSDL interface.

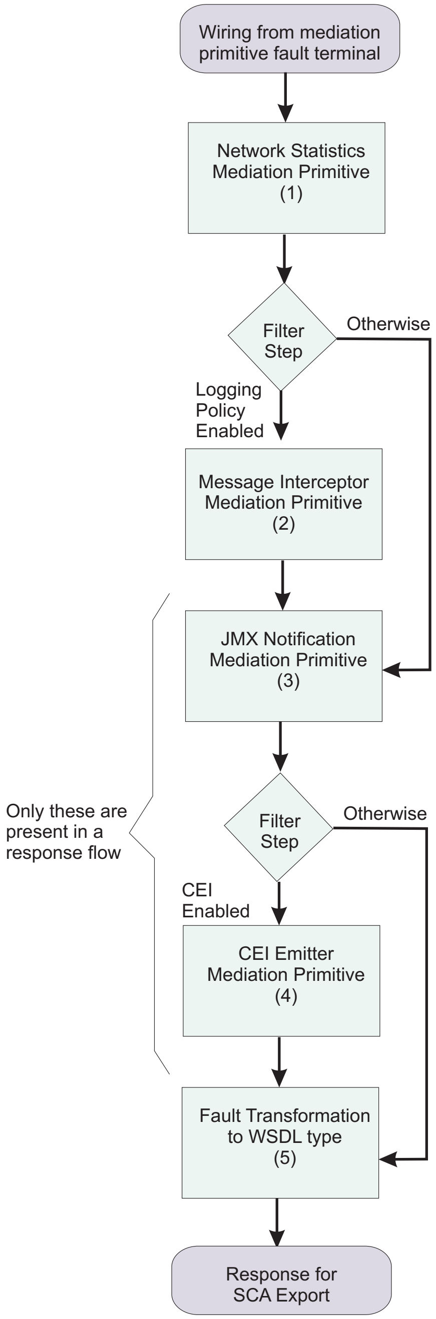

Wiring from mediation primitive fault terminal
Network Statistics Mediation Primitive (1)
Filter Step
Otherwise
Logging Policy Enabled
Message Interceptor Mediation Primitive (2)
JMX Notification Mediation Primitive (3)
Only these are present in a response flow
Filter Step
Otherwise
CEI Enabled
CEI Emitter Mediation Primitive (4)
Fault Transformation to WSDL type (5)
Response for SCA Export

*Figure 7. Access Gateway Fault Flow*

The default handling flow segment executes the following mediation primitives in sequence for a request flow. In a response flow, only the highlighted mediation primitives are included.

1. The Network Statistics mediation primitive is called to log statistics about the fault event.
2. If message capture is enabled by policy, the Message Interceptor mediation primitive logs the actual fault object being returned to the requester.
3. The JMX Notification mediation primitive emits a JMX notification, which contains information about the fault.
4. If there is a policy specifying that a common base event should be emitted, the CEI event emitter mediation primitive emits the common base event. This

enables the fault information to be picked up by external security or monitoring systems, such as Tivoli products.

5. An XSLT transformation is performed to create an appropriate WSDL-default fault from the SMO. This provides an appropriate fault response for the caller. The XSLT transformation mediation primitive is provided with the WebSphere ESB platform.

   **Note:** The XSLT transform file has changed in WebSphere ESB, which allows necessary porting to account for the change in the runtime.

   Any outgoing requests that originate from within a service implementation–for example, Web service notifications– must also pass through the TWSS Access Gateway. The request is subject to standard message processing logic. This allows for logging, calculation of network statistics, and other policy-driven actions. You can also choose to create a custom flow, which is slimmed down for outbound requests. The requests should indicate, within the SOAP header, that the Access Gateway is to perform as a Web service intermediary but not as a final destination for the request.

## SOAP attachments support

Access Gateway extends the WebSphere ESB platform to provide SOAP attachment support. The extension enables the Access Gateway to maintain the SOAP attachment data instead of discarding it.

### Mediation flows

IBM WebSphere Telecom Web Services Server (TWSS) provides APIs to access and process SOAP attachments within mediation primitives and the ability to propagate attachments through Service Component Architecture Web service imports.

### Performance

TWSS is optimized for SOAP attachments that are relatively small, ranging from 50 to 500 KB and averaging 150 to 300 KB. This allows the implementation to treat attachments as messages instead of streams.

If you anticipate having to handle larger attachments, you can avoid performance problems by enabling the Access Gateway to process SOAP attachments in memory, rather than using disk I/O operations. For details, see the topic *Configuring additional settings for the Access Gateway*.

## SOAP attachment code snippet

The following shows a snippet of code for retrieving an attachment from an inbound SOAP message with attachments.

SOAP attachment code snippet:

```
public void mediate(InputTerminal input, DataObject message)
    throws MediationConfigurationException, MediationBusinessException {

        ServiceMessageObject smo = (ServiceMessageObject)message;

        Attachments attachments = Attachments.INSTANCE;
        String contextID = attachments.getAttachmentContextIDForSMO(smo);
        Iterator it = attachments.getAllMetaData(contextID);
        // In this example, we assume there's one attachment and take the first one
        AttachmentMetaData metadata = (AttachmentMetaData)it.next();
        InputStream is = attachments.getAttachmentContent(contextID,
                metadata.getAttachmentID());
        // Read from stream


}
```

## Programming an SPM MBean script

The administrative scripting (wsadmin) allows for the creation of scripts which have the ability to interact with WebSphere Application Server objects, and are able to manipulate Service Policy Manager (SPM) user interfaces. The SPM runtime component provides a corresponding MBean interface for each of its Web services.

### Before you begin

The wsadmin tools supports Jacl and Jython scripting languages. Jacl is the default language. If you want to use the Jython scripting language, use the -lang option when you run the script or specify it in the `wsadmin.properties` file.

Ensure that the following are pre-configured or already installed:

- WebSphere Application Server Network Deployment version 7.0.0.1

  For more information, refer to the topic *WebSphere application server administration server configuration model* in the WebSphere Application Server information center.
- Service Policy Manager (SPM) Web service, installed and running

  For more information, refer to the topic *Installing Telecom Web Services Server base components* in this information center.
- Jython or Jacl (Jython is recommended)

  For more information, refer to the topic *Getting started with scripting* in the WebSphere Application Server information center, and click the links for Jython and Jacl in the Sub-topics list.

### Example

The following is a sample MBean program using the Jython scripting language. This script attempts to create requesters and remove requesters.

```
Script name: IMS_SPM_RequestersAdmin_MBean_sample.py

###########################################################################
# How to run this script:
# Copy this to IMS WAS bin directory and enter:
# ./wsadmin.sh -f IMS_SPM_RequestersAdmin_MBean_sample.py -lang jython
###########################################################################

import javax.management.ObjectName
import java.lang.System as sys
import pytwss.mbean.mbean_utils
import com.ibm.twss.spm.admin.policy
import com.ibm.twss.spm.admin.common.ServicePolicyException as SPE

# Get MBean type RequesterAdministration
spmReqAdmin=AdminControl.queryNames('WebSphere:type=RequesterAdministration,*')

if (spmReqAdmin == ""):
 print "can not get MBean type RequesterAdministration from WebSphere runtime. Make sure"
 print " 1) You have installed Service Policy Manager Runtime (SPM) application"
 print " 2) server and the SPM runtime application are started"
 sys.exit(0)
```

```
# Uncomment below line to see help on MBean if needed
#print Help.all(spmReqAdmin)

#List of Requester Definitions to be attempted
Requesters_list =[['Grp1', 'ALL', 1 , 'Grp1 Desc', 'RequesterGroup'],
   ['Grp2', 'ALL', 1 , 'Grp2 Desc', 'RequesterGroup'],
   ['Grp3', 'Grp2', 1 , 'Grp3 Desc', 'RequesterGroup'],
   ['Req1-1', 'Grp1', 1 , 'Req1-1 Desc', 'Requester'],
   ['Req1-2', 'Grp1', 1 , 'Req1-2 Desc', 'Requester'],
   ['Req3-1', 'Grp3', 1 , 'Req3-1 Desc', 'Requester'],
   ['Req1-1', 'Grp1', 1 , 'Req1-1 Desc', 'Requester'],
   ['Req1-1', 'Grp1', 1 , 'Req1-1 Desc', 'Requester'],
   ['Req1-1', 'DummyGroup', 1 , 'Req1-1 Desc', 'Requester']]


print "================= Attempting CreateRequesters ================="
i=0
for i in range(0, len(Requesters_list)):

 print " ======== Attempting record: ", Requesters_list[i], "========="

 #Instanciate RequestDefinition and set values
 ReqDef=com.ibm.twss.spm.admin.common.RequesterDefinition()
 ReqDef.setRequester(Requesters_list[i][0])
 ReqDef.setParentGroup(Requesters_list[i][1])
 ReqDef.setEnabled(Requesters_list[i][2])
 ReqDef.setDescription(Requesters_list[i][3])
 ReqType=com.ibm.twss.spm.admin.common.RequesterType.fromString(Requesters_list[i][4])
 ReqDef.setDefinitionType(ReqType)

 #Instanciate RequestDefinitionList
 ReqDefList = com.ibm.twss.spm.admin.common.RequesterDefinitionList()
 ReqDefList.setDefinition([ReqDef])

 #Instanciate CreateRequestersRequest and set RequestersDefinitionList
 CreReqReq=com.ibm.twss.spm.admin.req.CreateRequestersRequest()
 CreReqReq.setDefinitions(ReqDefList)

 #Instanciate CreateRequestersResponse
 CreReqRes=com.ibm.twss.spm.admin.req.CreateRequestersResponse()

 #Invoke the createRequesters operation
 CreReqRes = AdminControl.invoke_jmx(javax.management.ObjectName(spmReqAdmin), 'createRequesters',


 #Get any ServicePolicyExceptions
 SPE = ((CreReqRes.getFaultStatus()).getFaultStatus(0)).getFault()
 if SPE is not None:
  #Print ServicePolicyException ID and text
  print "  ", SPE.getText(), "\n"

 if SPE is None:
  print "  Successful"
 i=i+1


print "===================== Attempting RemoveRequesters ================="
Remove_list =[['Grp1'], ['Gr1'], ['Req1-1']]

i=0
for i in range(0, len(Remove_list)):

 print " ======== Attempting record: ", Remove_list[i], "========="

 RemReqList=com.ibm.twss.spm.admin.common.RequesterList()
 RemReqList.setRequester(Remove_list[i])
```

```
RemReqReq=com.ibm.twss.spm.admin.req.RemoveRequestersRequest()
RemReqReq.setRequesters(RemReqList)

RemReqRes=AdminControl.invoke_jmx(javax.management.ObjectName(spmReqAdmin), 'removeRequesters', [Re

#Get any ServicePolicyExceptions
SPE = ((RemReqRes.getFaultStatus()).getFaultStatus(0)).getFault()
if SPE is not None:
 #Print ServicePolicyException ID and text
 print "  ", SPE.getText(), "\n"

if SPE is None:
 print "  Successful"

i=i+1
```

## Migrating your applications from a previous release of Telecom Web Services Server

Legacy applications written to interact with the Parlay Connector may need to be modified before they will run in a Telecom Web Services Server (TWSS) version 7.0 environment. Applications written to the version 6.2 Web service implementations can run without being modified.

### Parlay-based applications

For application programs written to interact with a Parlay Connector using the functionality in TWSS version 6.2, the following modifications are required to make them work in a version 7.0 environment.

- Add the following application dependencies.

  ParlayConnectorEJBClient

  parlaymapclient.jar

- Remove the following application dependencies, if they exist. If they are separate projects, delete them.

  ParlayConnectorEJB

  ParlayWebConsole

  WssDataEJB

  WssRuntimeEJB

  parlay

  parlaymap

  runtime

- If the application uses a Java Object as a callback, convert it to Stateless Session EJB.
- Remove any references to the QueuedObjectEvent class, and refer instead to the QueuedObjectInvoker class. Methods in the QueuedObjectInvoker class call the appropriate methods in the ParlayConnectorRemoteBean remotely. The following new methods have been introduced to obtain queue location information:

  ParlayConnectorRemoteOperations remoteOperations = ParlayConnectorFactory.getRemoteConnector();

  String qLocation = remoteOperations.getQueueLocation();

  String[] locations = remoteOperations.getQueueLocations();

- Modify the application to accommodate the following changes to the QueuedObjectInvoker class:

  - Methods in the QueuedObjectInvoker class now generate RemoteException exceptions, so you should modify your application to handle these exceptions. (Application clients should not use QueuedObjectInvoker.post() and QueuedObjectInvoker.send() methods for peer-to-peer communication or to invoke asynchronous methods on stateful beans.)
  - Your application should invoke the new removeSession() method to clean up session objects stored on the Parlay Connector at the end of a session.
- Replace any calls to ParlayConnectorFactory.getConnector() with calls to ParlayConnectorFactory.getRemoteConnector().
- Remove any EJB references to the Application Manager.
- So that it can register for managed events with the Application Manager, the stateless session bean that implements the ManagedObjectOperations interface must have a JNDI name with the following format:

  `ejb/WAST/Application/ordinal/PC_name/app_name/interface_name`

  where

  *ordinal* is a number that specifies the order in which management beans are displayed in the Parlay console (the Parlay Connector is always assigned the number 1)

  *PC_name* is the name of the Parlay Connector

  *app_name* is the name of the application

  *interface_name* is the Remote Interface name of the bean

## Applications that integrate with the TWSS Web service implementations

Application programs that were based on the TWSS version 6.2 Web service implementations will run unchanged in a version 7.0 environment.

You may, however, choose to modify the applications to take advantages of new call flows and interfaces that have been introduced in version 7.0. For details, see *Integrating with Web service implementations* in this information center.

# Chapter 7. Troubleshooting WebSphere Telecom Web Services Server

WebSphere Telecom Web Services Server generates logs and messages to help you during problem determination. Logs store information to help you troubleshoot problems, and messages provide explanations and responses to help you resolve errors.

## Using ISA 4.0 add-ons to communicate with IBM Support

To help you communicate with IBM Support, an IBM Support Assistant (ISA) 4.0 product add-on is available on the Web for IBM WebSphere Telecom Web Services Server. You can install the add-ons for selected products and features using the ISA graphical user interface.

### About this task

You can open electronic service requests using the ISA add-ons. If you want to send log files associated with the service request, you must install and use the add-on for the version of WebSphere Application Server that you are running. It collects logs, trace files, and configuration information to send to IBM Support.

To install the product add-ons, perform the following steps:

1. Download and install ISA, using the instructions found on the IBM Support Assistant Web site.
2. Launch the IBM Support Assistant Workbench.
3. Click **Update** → **Find new** → **Product Add-ons**.
4. In the Product Add-ons window, select the ISA product add-ons you want to install. The add-ons are categorized by product family.
   a. Expand the **WebSphere** product family.
   b. Check one or more products for which you want to install add-ons.
   c. Click **Next**.
5. In the Tools Add-ons window, select any additional ISA add-ons you want to install. Then click **Next**.
6. Review the license information for the add-ons you have selected, and click **I accept the terms in the license agreements**.
7. Click **Next**.
8. Click **Finish**.
9. Restart the IBM Support Assistant when the installation has completed.

## Troubleshooting Service Platform components

The Service Platform components are deployed on a WebSphere Application Server Network Deployment platform to provide common service implementation functions. Use this information, along with your normal service support, to assist in tuning or troubleshooting Service Platform components.

You can also refer to the WebSphere Application Server information center for additional information specific to that product.

## Admission Control component Web service

This component keeps track of service usage information and can constrain the rate of requests accepted by the service, both at the server level and cluster level. When used in a cluster, each cluster member maintains a local hierarchy of rate limiting buckets that are used to calculate request admission. Reservations for additional rate capacity are made against each operation as more rate is needed. When additional rate has been reserved, the allocate rate is added to the operation rate limiting bucket and the parent service rate limiting bucket. Limits set at the server level constrain the rate of requests for all operations executed against that service. Limits set at the operation level limit the rate of requests for the particular operation. The rate limit for the service bucket is the sum of the rates of its child operation buckets.

### Exceed Operation limit recorded as ServiceAdmissionControlFault

In certain cases, there is no way to determine if a reservation was denied due to a service or operation limit being exceeded. This is because Admission Control component Web service performs the admission against its local view. The following is an example of a local operation reservation request resulting in a ServiceAdmissionControlFault. Token flow rates are expressed in tokens per second.

1. At the cluster level, the server limit is set at 15 tokens, which means a local service can send a reservation request for up to 15 tokens. The cluster operation limit is set at 5, which means a local operation can send a reservation request for up to 5 tokens (per second). The Cluster service remaining is 15 tokens, because no tokens have yet been reserved. The cluster operation remaining is 5 tokens, because no tokens have yet been reserved. The following table shows the initial token count:

*Table 39. Initial number of tokens at the beginning of the operation.*

| Cluster service remaining | Cluster operation remaining | Local service limit (max 15) | Local operation limit (max 5) |
|---|---|---|---|
| 15 | 5 | 0 | 0 |

2. At the local view, the operation requests five tokens. The local reserved service limit, which is the sum of the reserved limits for its child operations, is set to five tokens. Note that the operation has requested the maximum number of tokens it can reserve. The cluster still has ten tokens remaining that could be requested by the service, but zero tokens are available for reservation by the operation.

*Table 40. Number of tokens after the initial reservation.*

| Cluster service remaining | Cluster operation remaining | Local service limit (max 15) | Local operation limit (max 5) |
|---|---|---|---|
| 10 | 0 | 5 | 5 |

3. At the local view, the operation requests an additional five tokens within the same time span (one second) as the first request. It receives a reserve amount of zero from the cluster as each operation is limited to five tokens per second. The operation is rejected and no change is made to the reservation amount.
4. The Admission Control component Web service returns a ServiceAdmissionControlFault because it cannot determine whether the reservation for tokens was rejected as a result of an operation level limit or a

service level limit being exceeded. No change is made to the reservation amount. The node member receiving the rejected reservation request enters a silence period.

Verification of whether a request can be admitted is performed using the local view. A reservation request may be denied at any time due to lack of sufficient rate at the service or operation level. When a node member receives a rejected reservation request, it will enter a silence period. This silence period will suppress additional reservation requests, reducing inter-cluster traffic and allowing the network to settle before attempting additional reservation. In steady state, this will minimize the inter-cluster messaging required.

### Address Masking component Web service

If address masking does not work as expected, verify that you configured the masking and shadowing operations correctly. For example, are the right number of digits specified for shadowing? If masking expiry is used, has the specified time interval elapsed?

## Character encoding for the TWSS Administration Console

In non-U.S. locales, it might be necessary to set character encoding to UTF-8 when configuring the TWSS Administration Console.

The TWSS Administration Console is an extension to the WebSphere Integrated Solutions Console.

If you observe question marks either before or after strings like `TWSS Admin console`, configure the console such that it uses UTF-8 encoding.

## Configuring a secondary Parlay gateway for failover

When you configure a secondary Parlay gateway for failover, you must ensure that the service IDs for the secondary gateway are different from those of the primary gateway. Ensure that each gateway will create unique session IDs and assignment IDs for the Parlay interactions.

The Parlay-based Web service implementations use the gateway session IDs and assignment IDs as references to track requests sent to the gateway.

**Note:** If the primary gateway fails and the secondary, or backup, gateway assumes the workload, the session IDs and assignment IDs for each gateway must be unique or the following exception is generated. `SOAM1578E: Not able to create the record of the session object.`

## Troubleshooting WID

While installing the WID plugins for Telecom Web Services Server 7.0 on WID, the WID plugin installer checks for the integrated test server. If you do not opt for the integrated test server install, you will receive the error message *"Unsupported WID home directory selected"*.

### About this task

The error message occurs when you do not select the integrated test server installation, even though you have a supported WID version already installed. Your installation process will subsequently be aborted.

To resolve this issue, perform either option A or option B below:

**Option A**:

1. Before installing the Telecom Web Services ServerWID plugin, you can install the integrated test server that comes with WID.

**Option B**:

1. Before installing Telecom Web Services Server WID plugin, you can create an empty directory called **bi_v62** (for 6.2.x WID) or **bi_v61** (for 6.1.x WID), under the directory `WID_HOME/runtimes/`.

## Monitoring log messages

WebSphere Telecom Web Services Server can write system messages to several general purpose logs. Logging provides information about important lifecycle events, warnings, and errors that should be addressed by an administrator.

By default, WebSphere Telecom Web Services Server logs its messages to the WebSphere Application Server JVM log (`SystemOut.log`) and its trace messages to the WebSphere Application Server trace log (`trace.log`). Both log files are located in the `logs` directory:

AIX *was_profile_root*/logs/*server_name*

Linux *was_profile_root*/logs/*server_name*

*was_profile_root*/logs/*server_name*

**Note:** *was_profile_root* is the directory for a WebSphere Application Server Network Deployment profile called *profile_name*. By default, this directory is:

AIX /usr/IBM/WebSphere/AppServer/profiles/*profile_name*

Linux /opt/IBM/WebSphere/AppServer/profiles/*profile_name*

/opt/IBM/WebSphere/AppServer/profiles/*profile_name*

In a standalone environment, *profile_name* is the name of the application server profile. In a clustered environment, *profile_name* is the name of a federated node profile.

Each error, warning, or informational log message should include a message code which is used to identify the message. Additionally, each message can be identified by the date, timestamp, thread number, and severity. For example:

```
[5/5/09 18:38:03:598 EDT] 0000002d AbstractServi I
com.ibm.twss.parlayx21.termstatus.AbstractServiceBindingImpl
getSipURIFromTarget SOAX3511I:
SOAContextImpl_9.42.126.119_1146868683342_2047515057: Presence_IMS:
Unable to retrieve presence information for target http://hostname.myco.com.
Unable to parse target as a SIP URI: retval.getScheme() = http
```

Comprehensive information about working with message logs may be found in the WebSphere Application Server Network Deployment information center.

## Viewing and modifying logs

Use the Integrated Solutions Console to specify how data is logged, where the log data is stored, and the output format to use for log data.

### About this task

You can modify the general properties of each log, which specifies the output type or location of the log. Use the following steps to adjust the properties for each log type:

1. Log in to the Integrated Solutions Console:
   a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

      Where:

      *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

      *port* is the secured port used to access the console. The default port is *9043*.

      **Note:** The default unsecured port is *9060*. If you use 9060, you must have "http" instead of "https" in the URL.
   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Servers** → **Server Types** → **WebSphere application servers**.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Application servers**.
3. Click the name of the server you want to manage.
4. Under Troubleshooting, click **Logging and Tracing**.
5. Click one of the log types. Then click the Configuration tab to make a static change to the system log configuration, or click the Runtime tab to change the configuration dynamically.

   **Note:** Separate logs for each log type exist for all Java virtual machines (JVMs) on a node, including all application servers and their node agent, if present, as well as for a deployment manager in its own logs directory.

   Here is a list of the available log types:

| Option | Description |
|---|---|
| **Diagnostic Trace** | Provides information in the `trace.log` about how the WebSphere Application Server components run. |
| **JVM Logs** | Used to view and modify the settings for the Java Virtual Machine (JVM). The `System.out` log (`SystemOut.log`) is used to monitor the health of the WebSphere Application Server. The `System.err` log (`SystemErr.log`) contains exception stack trace information used to perform problem analysis. |

| Option | Description |
|---|---|
| **Process Logs** | Created when redirecting the standard out and standard error streams of a process to independent log files, the `native_stdout.log` and `native_stderr.log`, respectively. |
| **IBM Service Logs** | Also known as the activity log. Records the WebSphere Application Server messages that are written to the `System.out` stream and special messages that contain extended service information that you can use to analyze problems. |
| **Change Log Detail Levels** | Controls which events are processed by Java logging, by using log levels. You can assign logging levels to individual trace loggers or to trace groups. (Trace loggers and groups are listed in the topic *Trace loggers*.) |

6. When you are finished making your changes, click **Apply**.
7. Click **OK**.
8. Click **Save** to save changes to the master configuration.
9. Optional: If you made a static change to the configuration, restart the application for your changes to take effect.

### Results

After your configuration changes take effect, you will be able to view log data in the locations, and in the output formats, that you have specified. Note that certain logging settings can affect the performance of your system.

## Enabling trace

Trace logs show trace events such as function entries and exits, component events, and debugging activities. Use the administration console to enable trace for a process.

### About this task

You can configure the WebSphere Telecom Web Services Server to start in a trace-enabled state by setting the appropriate configuration properties.

You can control how much detail each logger records by adjusting the log level details. Because the loggers are grouped hierarchically, setting the trace level on one logger also sets all subsequent loggers to the same level. Altering the tracing levels impact the performance of the system.

Enable and configure trace by completing the following steps:

1. Log in to the Integrated Solutions Console:
    a. Open a browser and navigate to the following URL: https://*host_name*:*port*/ibm/console.

       Where:

       *host_name* is the fully qualified host name of the server where the application or the network deployment manager is deployed.

       *port* is the secured port used to access the console. The default port is *9043*.

**Note:** The default unsecured port is *9060*. If you use 9060, you must have ″http″ instead of ″https″ in the URL.

   b. Enter an administrator user ID and password. (Omit the password if security is not enabled.)
   c. Click **Log in**.
2. In the navigation pane, click **Servers** → **Server Types** → **WebSphere application servers**.

   **Note:** If you are using WebSphere Application Server version 6.1.0.x, reach this window by clicking **Servers** → **Application servers**.
3. Click the name of the server you want to manage.
4. Click **Troubleshooting** → **Logging and Tracing**.
5. Click **Diagnostic Trace Service**.
6. Configure your trace options:
   a. Display the Runtime tab.
   b. To disable tracing, select **File** and then select **None**.

      **Note:** If you are using WebSphere Application Server version 6.1.0.x, disable tracing by selecting **Enable log** and then substituting **disabled** in place of **enabled**.
   c. Click **Change Log Level Details**.
   d. Click **Components** to view all loggers for the individual components.
   e. Click **+** to show the *children* of the logger.
   f. Click ***logger_name*** to change the log details. To enable tracing on specific components of WebSphere Telecom Web Services Server, click one of these logger groups:
      com.ibm.soa.common.*
      com.ibm.soa.parlayx21.*
      com.ibm.soa.sp.*
      com.ibm.soa.*
      com.ibm.twss.ag.*
      com.ibm.twss.parlayx21.*
      com.ibm.twss.spm.*
      com.ibm.twss.*
      com.ibm.mds.*
      com.ibm.wast.*
   g. Choose the appropriate level of tracing.

      **Remember:** When you change the level for a logger, the change is propagated to the children of the logger.
      For additional information regarding trace levels, click **?** in the title bar of the panel to open the help page.
7. Click **OK**.
8. Click **Save**.

## Results

The specified traces are enabled for the current server session. To make the changes permanent, use the Configuration tab rather than the Runtime tab when you configure the trace options. Note that when you use the Configuration tab, you will need to restart the server for your changes to take effect.

## Selecting trace loggers

The level of tracing is determined by the log level details you select for the loggers. Loggers are organized hierarchically. The children of the logger will inherit the parent log level by default, but it can be changed by defining the level of tracing on each specific logger.

To control the trace level for the Trust Association Interceptor, use the options on the `com.ibm.imsconnector.*` trace group. For more specific levels of tracing, use the following trace groups, which are relevant to the Trust Association Interceptor:

*Table 41. TAI trace groups and trace loggers*

| **Trace group** | **Trace loggers** |
|---|---|
| com.ibm.imsconnector.tai.* | *BaseIMSInterceptor<br>*HttpInterceptor<br>*SipInterceptor |

To control the trace level for all subcomponents of the Telecom Web Services Server component and its subcomponents, use the options on the `com.ibm.twss.*`, `com.ibm.mds.*`, and `com.ibm.wast.*` trace groups. For more specific levels of tracing, use the following trace groups, which are relevant to Telecom Web Services Server:

*Table 42. TWSS trace group and trace loggers*

| **Trace groups** | **Trace loggers** |
|---|---|
| com.ibm.soa.common.* | *mbean.*<br>*management.* |
| com.ibm.soa.sp.* | *admctl.*<br>*fltalm.*<br>*netres.*<br>*privacy.*<br>*pxnotify.*<br>*trafficsh.*<br>*userec.*<br>*notifymgmt* |
| com.ibm.twss.ag.* | *mediation.* |
| com.ibm.twss.parlayx21.tl.mlp.* | *ctrl.*<br>*mlp.util.*<br>*mlp.conn.*<br>*mlp.mbean.* |
| com.ibm.twss.parlayx21.* | *tpc.*<br>*ch.*<br>*cn.*<br>*callcontrol.* |
| com.ibm.soa.parlayx21.ims.* | *call_notification.*<br>*group.*<br>*payment.*<br>*presence.*<br>*termstatus.*<br>*thirdparty.* |
| com.ibm.soa.common.* | *management.*<br>*mbean.* |

*Table 42. TWSS trace group and trace loggers (continued)*

| Trace groups | Trace loggers |
|---|---|
| com.ibm.mds.* | *adpt.*<br>*adpt.mms.mm7.*<br>*adpt.mms.http.mm7.*<br>*adpt.sms.smpp..*<br>*comm.*<br>*loc.*<br>*mms.*<br>*sms.*<br>*stat.*<br>*util.* |
| com.ibm.wast.* | *license.*<br>*parlay.* |

**AG attachments:**

- com.ibm.websphere.sca.soap.attachments.*
- com.ibm.ws.sca.soap.attachments.*

**AG handlers:**

- com.ibm.soa.esb.global.handlers.*
- com.ibm.sca.connections.handlers.*
- com.ibm.ws.sca.soap.attachments.handlers.*

### Additional tracing information

You can also trace a SIP container or CEI. For more detailed information about tracing a SIP container or CEI, refer to the WebSphere Application Server Network Deployment Information Center.

> To control the trace level for the SIP container, use the options on the com.ibm.ws.sip.* trace group.
>
> To control the trace level for the CEI emitter, use the options on the com.ibm.events.emitter.*

There are additional trace options for WebSphere Enterprise Service Bus. For more detailed information about tracing ESB, refer to the WebSphere Enterprise Service Bus Information Center.

> To control the trace level for the ESB Service Component Architecture, you can use **SCA = all**.

# Messages

A message explains a problem and suggests a user action. In addition, each message ID includes a component ID, a number, and a letter that indicates the type of message: Informational, warning, or error.

## Message key

Each sub component has a unique message identifier to help you determine the origin of the message.

## Standard format

The standard message format is: `AAAANNNNS`

- `AAAA` represents the component identifier (typically four or five characters).
- `NNNN` represents a four digit identifier.
- `S` represents the type of messages. There are three message types:
  - `I` represents informational messages.
  - `W` represents warning messages.
  - `E` represents error messages.

## WebSphere Telecom Web Services Server messages

The following message identifiers are used for WebSphere Telecom Web Services Server:

*Table 43. WebSphere Telecom Web Services Server*

| Component identifier | Component description |
|---|---|
| SOAC | Access Gateway and Service Platform components messages. |
| SOAM | Messages for the Parlay-based and Direct Connect-based Web service implementations, including JCA adapter messages. |
| SOAS | Service Platform components messages |
| SOAX | Messages for the SIP/IMS-based Web service implementations, for Parlay X Address List Manager over XCAP, and for Parlay X Payment (PostPaid). |
| TAS | Parlay Connector messages |

## Trust Association Interceptor messages

The following message identifier is used for Trust Association Interceptor:

*Table 44. Trust Association Interceptor*

| Component identifier | Component description |
|---|---|
| DHAT | TAI messages |

# Messages_SOAC_ESB_en_US

This section documents the set of messages that the ESB component can issue.

**SOAC4001E:{0}{1} {2} not available from System.getProperty**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the custom property that was not found defined for the Application Server.

**User Response:**

- Ensure that the custom property is added to the Application Server's JMV process environement.

**SOAC4002E:{0}{1} Could not reach {2} Sub-system: {3}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the Sub-system that cannot be reached. *{3}* is the message from the exception. An exception occured while attempting to send a request to the specified Sub-system.

**User Response:**

- None: review exception message for possible causes.

**SOAC4003E:{0}{1} Invalid URL set for {2} Sub-system: {3}**
**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the name of the Sub-system that cannot be reached. *{3}* is the value of the URL that was used to reach the specified Sub-system. An invalid URL was set for the specified Sub-system.

**User Response:**

- Verify the URL value for the specified Sub-system.

**SOAC4004E:{0}{1} IOException when parsing SMO: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the message from the io exception. An IO exception occurred while parsing the Service Message Object.

**User Response:**

- None: review exception message for possible causes.

**SOAC4005E:{0}{1} ParserConfigurationException when parsing SMO: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the message from the parser configuration exception. An parser configuration exception occurred while parsing the Service Message Object.

**User Response:**

- None: review exception message for possible causes.

**SOAC4006E:{0}{1} SAXException when parsing SMO: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the message from the SAX parser exception. An SAX parser exception occurred while parsing the Service Message Object.

**User Response:**

- None: review exception message for possible causes.

**SOAC4007E:{0}{1} Policy Exception when retrieving policies: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the message from the policy exception. An policy exception occurred while retrieving policy data.

**User Response:**

- None: review exception message for possible causes.

**SOAC4008E:{0}{1} Service Exception when retrieving policies: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the message from the service exception. An service exception occurred while retrieving policy data.

**User Response:**

- None: review exception message for possible causes.

**SOAC4009E:{0}{1} Exception: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the message from the exception. An exception occurred.

**User Response:**

- None: review exception message for possible causes.

**SOAC4010E:{0}{1} SOAP header twssHeader was not found in SMO**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The twssHeader soap header was not found in the Service Message Object.

**User Response:**

- None: possibly a programming error.

**SOAC4011E:{0}{1} SOAP element policies was not found in twssHeader**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The policies element was not found in the twssHeader soap header.

**User Response:**

- None: possibly a programming error.

**SOAC4012E:{0}{1} GlobalTransactionID was not set in twssHeader**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The GlobalTransactionID element was not found in the twssHeader soap header.

**User Response:**

- None: possibly a programming error.

**SOAC4013E:{0}{1} Unable to send CEI event due to error: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the exception message issued. An exception occurred attempting to send a CEI event.

**User Response:**

- None. Review the exception message for possible causes.

**SOAC4014E:{0}{1} Unable to get Initial Context due to: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the exception message issued. An exception occurred attempting to get the intial context data.

**User Response:**

- 
- None: possibly a programming error.

**SOAC4015E:{0}{1} Unable to get emitter due to: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the exception message issued. An exception occurred attempting to get cei emitter.

**User Response:**

- None: possibly a programming error.

**SOAC4016E:{0}{1} Unable to get Event Factory due to: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the exception message issued. An exception occurred attempting to get Event Factory.

**User Response:**

- None: possibly a programming error.

**SOAC4017E:{0}{1} Unable to complete event due to: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the exception message issued. An exception occurred attempting to send cei event.

**User Response:**

- None: review exception message for possible causes.

**SOAC4018E:{0}{1} Unable to get component location due to: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the exception message issued. An exception occurred attempting to get component location.

**User Response:**

- None: review exception message for possible causes.

**SOAC4019E:{0}{1} Could not form URI from address string: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the address string that could not be formed into a URI. An error occurred attempting to form URI from address string provided.

**User Response:**

- None: review address string and validate content.

**SOAC4020E:{0}{1} Unable to obtain Group proxy**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. An error occurred attempting to get Group proxcy.

**User Response:**

- None: possibly a programming error.

**SOAC4021E:{0}{1} Unable to access work area partition manager: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the work area partition manager. An error occurred attempting to access the Application Server work area partition manager.

**User Response:**

- None: possibly a programming error.

**SOAC4022E:{0}{1} work area partition not found: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the work area partition. An error occurred attempting to access the Application Server work area partition.

**User Response:**

- None: possibly a programming error.

**SOAC4023E:{0}{1} No value was specified for the {2} element in TWSSheaders**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the soap element in the TWSSheaders soap header that contained no value The named soap element was present in the TWSSheaders soap header but had no value specified.

**User Response:**

- None: possibly a programming error.

**SOAC4024E:{0}{1} Unable to obtain database connection for: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the datasource attempting to obtain connection for An error occurred attempting to obtain a database connection for the specified datasource.

**User Response:**

- Verify the datasource name is correct and is properly configured.

**SOAC4025E:{0}{1} Insert into the {2} table failed. See trace log.**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the database table attempting to insert data into An error occurred inserting data into the specified table.

**User Response:**

- None: review JVM log messages for possible causes.

**SOAC4026W:{0}{1} Could not close database connection due to: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the warning message issued. An warning occurred attempting to close the database.

**User Response:**

- None: possibly a programming error.

**SOAC4027E:{0}{1} Could not access datasource {2} due to: {3}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the datasource attempting to access. *{3}* is the exception message issued. An exception occurred attempting to access the specified datasource.

**User Response:**

- None: review JVM log messages for possible causes.

**SOAC4028E:{0}{1} Could not get connection to datasource {2} due to: {3}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the datasource attempting to get connection for. *{3}* is the exception message issued. An exception occurred attempting to get connection for the specified datasource.

**User Response:**

- None: review JVM log messages for possible causes.

**SOAC4029E:{0}{1} Failed writing to the repository due to: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the exception message issued. An error occurred writing to the database.

**User Response:**

- None: review JVM log messages for possible causes.

**SOAC4030E:{0}{1} SQL statements not loaded, could not locate properties file: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the properties file containing the SQL statements. An error occurred while attempting to locate the specified properities file.

**User Response:**

- None. possibly a programming error.

**SOAC4031E:{0}{1} Could not find {2} SQL statement: {3}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the database type (DB2, Oracle, ...etc.) *{3}* is the name of the SQL statement trying to load An error occurred while attempting to locate the SQL statement for the specified database type.

**User Response:**

- None. possibly a programming error.

**SOAC4032E:{0}{1} Default database value used, could not get the type due to: {2}**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the error message issued An error occurred while attempting to set the database type, default DB2 will be used.

**User Response:**

- None. possibly a programming error.

**SOAC4033E:{0}{1} requesterID was not set in twssHeader**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The requesterID soap element was not present in the TWSSheaders soap header.

**User Response:**

- None. possibly a programming error.

**SOAC4034E:{0}{1} Anonymous Access not authorized**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The request was received with no requesteriD specified in the TWSSheaders soap header or the value was set to ″Anonymous″.

**User Response:**

- Specify a different requesterID or set policies to allow for Anonymous access.

**SOAC4035E:{0}{1} Requester was not authorized to access service**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The requesterID specified in the request was not authorized to access the specified service.

**User Response:**

- None. Access is granted by setting up policies for the requesterID and the requested service.

**SOAC4036E:{0}{1} Requester was not authorized to access service operation**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The requesterID specified in the request was not authorized to access the specified service operation.

**User Response:**

- None. Access is granted by setting up policies for the requesterID and the requested service operation.

**SOAC4037E:{0}{1} SOAP headers was missing from SMO**

**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The Service Message Object contained no SOAP headers.

**User Response:**

- None. possibly a programming error.

**SOAC4038E:{0}{1} No Service Endpoint present in SOAP header policies**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The policy data present in the TWSS SOAP header did not contain the service endpoint policy: **service.Endpoint**.

**User Response:**

- Add the service.Endpoint policy to the set of policies in the Service Policy Manager and retry the transaction. For detailed instructions on creating this service policy, see the InfoCenter under Service Policy AdministrationThe following example shows the required parameters for creating this policy where the user named user1 is given permission to invoke an implementation of the web service Third Party Call, operation makeCall: **Requester**: user1 **Service**: http://www.csapi.org/wsdl/parlayx/third_party_call/v2_3/interface (found in the service .wsdl file) **Operation**: makeCall **Name**: service.Endpoint **Value**: http://ibm.com:9080/soa/parlayx21/ThirdPartyCall/IMS/services/ThirdPartyCall (Note: the service.Endpoint value is for the back-end web service, not the published ESB mediation flow endpoint. This value can be found in the Administrative Console of the server where the web service is installed.)

**SOAC4039W:{0}{1} Unable to find handler data object in work area: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the Application Server partition work area. The partition work area did not contain the required handler data object.

**User Response:**

- None. possibly a programming error.

**SOAC4040E:{0}{1} Limit exceeded for service operation: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the service operation. Processing the request would cause the defined Service Level Agreement (SLA) to be exceeded for the service operation.

**User Response:**

- None. Retry the request. If problem persist, try increasing the SLA limit that was exceeded.

**SOAC4041E:{0}{1} Limit exceeded for service: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the service. Processing the request would cause the defined Service Level Agreement (SLA) to be exceeded for the service.

**User Response:**

- None. Retry the request. If problem persist, try increasing the SLA limit that was exceeded.

**SOAC4042E:{0}{1} Limit exceeded for requester: {2}**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. *{2}* is the name of the requester. Processing the request would cause the defined Service Level Agreement (SLA) to be exceeded for the requester.

**User Response:**

- None. Retry the request. If problem persist, try increasing the SLA limit that was exceeded.

**SOAC4043E:{0}{1} Incomplete parameters, cannot retrieve policies**
**Explanation:**

*{0}* is the transaction ID for the request. *{1}* is the name of the component that issued the message. The request did not contain enough information to be able to retrieve the required policy data.

**User Response:**

- Ensure the request contains the requester, service, and service operation data.

**SOAC4044E:{0}{1} Attempted to set invalid interval: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the invalid interval value Attempted to set invalid interval.

**User Response:**

- None. possibly a programming error.

**SOAC4045E:{0}{1} Attempted to set invalid max entries: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the invalid max entries value Attempted to set invalid max entries.

**User Response:**

- None. possibly a programming error.

**SOAC4046E:{0}{1} Attempted to set limit on (null) bucket hierarchy**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. Attempted to set limit on null bucket hierarchy.

**User Response:**

- None. possibly a programming error.

**SOAC4047E:{0}{1} Attempted to set invalid limit: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the invalid max entries value Attempted to set invalid limit.

**User Response:**

- None. possibly a programming error.

**SOAC4048E:{0}{1} Attempted to get limit on (null) bucket hierarchy**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. Attempted to get limit on null bucket hierarchy.

**User Response:**

- None. possibly a programming error.

**SOAC4049E:{0}{1} Attempted to traverse path with (null) hierarchy levels**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. Attempted to traverse path with null hierarchy levels

**User Response:**

- None. possibly a programming error.

**SOAC4050E:{0}{1} (null) {2} element in hierarchy.**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the index value into the levelNames String array The levelNames entry in the String array at the specified index was null

**User Response:**

- None. possibly a programming error.

**SOAC4051E:{0}{1} Error accessing SOA HA SLA reservation singleton instance: (null)**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. Error accessing SOA HA SLA reservation singleton instance. The instance was null.

**User Response:**

- None. possibly a programming error.

**SOAC4052E:{0}{1} Attempted to get usage estimator for (null) requester**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. Attempted to get usage estimator but the requester was null.

**User Response:**

- None. possibly a programming error.

**SOAC4053E:{0}{1} Attempted to verify {2} SLA for (null) requester**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the controller type (local,cluster) Attempted to verify SLA but the requester was null.

**User Response:**

- None. possibly a programming error.

**SOAC4054E:{0}{1} Attempted to verify {2} SLA for (null) service**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the controller type (local,cluster) Attempted to verify SLA but the service was null.

**User Response:**

- None. possibly a programming error.

**SOAC4055E:{0}{1} Attempted to verify {2} SLA for (null) operation**

**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the controller type (local,cluster) Attempted to verify SLA but the operation was null.

**User Response:**

- None. possibly a programming error.

**SOAC4056E:{0}{1} Attempted to verify SLA for invalid requester limit: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the requester limit value Attempted to verify SLA but the requester limit was invalid.

**User Response:**

- None. possibly a programming error.

**SOAC4057E:{0}{1} Attempted to verify SLA for invalid service limit: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the service limit value Attempted to verify SLA but the service limit was invalid.

**User Response:**

- None. possibly a programming error.

**SOAC4058E:{0}{1} Attempted to verify SLA for invalid operation limit: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the operation limit value Attempted to verify SLA but the operation limit was invalid.

**User Response:**

- None. possibly a programming error.

**SOAC4059E:{0}{1} Attempted to verify SLA for invalid request amount: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the request amount value Attempted to verify SLA but the request amount was invalid.

**User Response:**

- None. possibly a programming error.

**SOAC4060E:{0}{1} Attempted to set invalid keep alive interval: {2}**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. *{2}* is the keep alive value Attempted to set invalid keep alive interval.

**User Response:**

- None. possibly a programming error.

**SOAC4061E:{0}{1} Message body missing from SMO**
**Explanation:**

*{0}* n/a *{1}* is the name of the component that issued the message. Message body missing from SMO.

**User Response:**

- None. possibly a programming error.

## Messages_SOAC_en_US

This section documents the set of common messages that the ESB and Service Platform components can issue.

**SOAC0028E:{0}{1} An unexpected exception occurred. [{2}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the message. *{2}* is the exception description. An unexpected exception occurred.

**User Response:**

- Review the exception description to identify the problem.
- Review the fault records to identify any errors.

**SOAC0029E:{0}{1} An unexpected event occurred [{2}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the message. *{2}* is the event description. An unexpected event occurred.

**User Response:**

- Review the event description to identify the problem.
- Review the fault records to identify any errors.

**SOAC0032W:{0}{1} An unexpected exception occurred. Processing will continue. [{2}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the message. *{2}* is the exception description. An unexpected exception occurred, however processing will continue.

**User Response:**

- Review the exception description to identify the problem.
- Review the fault records to identify any errors.

**SOAC0044I:{0}{1} During SOAContext initialization, received exception {2} with message {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the message. *{2}* is the class name of the exception. *{3}* is the localized message from the exception. An exception occurred while trying to initialize the SOAContext.

**User Response:**

- Review the exception description to identify the problem.
- Review the fault records to identify any errors.

**WebSphere Telecom Web services Server**

**SOAC0054I:{0}{1} Fault {2} of severity {3} originated from {4}. Fault message is [{5}]. Fault detail is [{6}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the fault. *{2}* is the fault code that occurred. *{3}* is the severity of the fault. *{4}* is the source of the fault. *{5}* is

the message describing the fault. *{6}* is additional detail that may help in resolving the fault. The named component recorded a fault via the Faults and Alarms common component.

**User Response:**

- See the documentation for the given fault code for more information.

**SOAC0055W:{0}{1} Fault {2} of severity {3} originated from {4}. Fault message is [{5}]. Fault detail is [{6}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the fault. *{2}* is the fault code that occurred. *{3}* is the severity of the fault. *{4}* is the source of the fault. *{5}* is the message describing the fault. *{6}* is additional detail that may help in resolving the fault. The named component recorded a fault via the Faults and Alarms common component.

**User Response:**

- See the documentation for the given fault code for more information.

**SOAC0056E:{0}{1} Fault {2} of severity {3} originated from {4}. Fault message is [{5}]. Fault detail is [{6}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the fault. *{2}* is the fault code that occurred. *{3}* is the severity of the fault. *{4}* is the source of the fault. *{5}* is the message describing the fault. *{6}* is additional detail that may help in resolving the fault. The named component recorded a fault via the Faults and Alarms common component.

**User Response:**

- See the documentation for the given fault code for more information.

**SOAC0057I:{0}{1} Alarm {2} of type {3} originated from {4} on host {5}. Alarm message is [{6}]. Alarm detail is [{7}]. Alarm suggested action is [{8}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the alarm. *{2}* is the alarm code that occurred. *{3}* is the type of the alarm. *{4}* is the source of the alarm. *{5}* is the host where the alarm originated. *{6}* is the message describing the alarm. *{7}* is additional detail that may help in resolving the fault. *{8}* is an action that may resolve the alarm. The named component recorded an alarm via the Faults and Alarms common component.

**User Response:**

- See the documentation for the given alarm code for more information.

**SOAC0058W:{0}{1} Alarm {2} of type {3} originated from {4} on host {5}. Alarm message is [{6}]. Alarm detail is [{7}]. Alarm suggested action is [{8}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the alarm. *{2}* is the alarm code that occurred. *{3}* is the type of the alarm. *{4}* is the source of the alarm. *{5}* is the host where the alarm originated. *{6}* is the message describing the alarm. *{7}* is additional detail that may help in resolving the fault. *{8}* is an action that may resolve the alarm. The named component recorded an alarm via the Faults and Alarms common component.

**User Response:**

- See the documentation for the given alarm code for more information.

**SOAC0059E:{0}{1} Alarm {2} of type {3} originated from {4} on host {5}. Alarm message is [{6}]. Alarm detail is [{7}]. Alarm suggested action is [{8}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the alarm. *{2}* is the alarm code that occurred. *{3}* is the type of the alarm. *{4}* is the source of the alarm. *{5}* is the host where the alarm originated. *{6}* is the message describing the alarm. *{7}* is additional detail that may help in resolving the fault. *{8}* is an action that may resolve the alarm. The named component recorded an alarm via the Faults and Alarms common component.

**User Response:**

- See the documentation for the given alarm code for more information.

**SOAC7500I: Located mbean '%(1)s' -> %(2)s**

**Explanation:**

%*(name)* is the name of the bean. %*(bean)* is the bean name. The location status of the MBean.

**User Response:**

- No action is needed.

**SOAC7501W: JMX error: type=%(1)s,exception=%(2)s**

**Explanation:**

%*(type)* is the type of the MBean. %*(ex)* is the exception that occurred. The JMX Java Management Extension operation encountered an error.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7502E: Retry count exceeded: %(1)s**

**Explanation:**

%*(cnt)* is the retry count. The retry count was exceeded, which means that the expected MBean can not be connected to.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7503I: Registering service implementation: service=[%(1)s],implementation=[%(2)s],enabled=[%(3)s],description=[%(4)s]**

**Explanation:**

%*(svc)* is the service. %*(impl)* is the service implementation. %*(en)* is the enabled state. %*(desc)* is the description. The service implementation was registered.

**User Response:**

- No action is needed.

**SOAC7507I: Service implementation successfully registered.**

**Explanation:**

The service implementation was registered.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7508E: Service implementation not registered. Either it has already been registered or an error occurred querying existing implementations.**

**Explanation:**

The service implementation failed to register successfully.

**User Response:**

- No action is needed.

**SOAC7509I: Creating policies in the Service Policy Manager...**

**Explanation:**

The policies are created.

**User Response:**

- No action is needed.

**SOAC7510I: Received results:**

**Explanation:**

The results are displayed.

**User Response:**

- No action is needed.

**[%(1)s] -> %(2)s**

**Explanation:**

*%(pol)* is the policy name. *%(res)* is the result. The policy value is displayed.

**User Response:**

- No action is needed.

**SOAC7512E: Error occurred - See server log for details.**

**Explanation:**

An error occurred that was not expected.

**User Response:**

- Review the configuration data provided to the script.

**Success**

**Explanation:**

The operation was successful.

**User Response:**

- No action is needed.

**req=%(1)s,svc=%(2)s,op=%(3)s,name=%(4)s,value=%(5)s,type=%(6)s**

**Explanation:**

*%(req)* is the requester. *%(svc)* is the service. *%(op)* is the operation. *%(name)* is the policy name. *%(val)* is the value. *%(type)* is the type. the policy information is displayed.

**User Response:**

- No action is needed.

**SOAC7515E: Failure: %(1)s %(2)s %(3)s**

**Explanation:**

*%(msgid)* the message id. The operation failed.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7516E: Error registering service implementation: [%(1)s]**

**Explanation:**

*%(msgid)* is the message id. The service implementation was not registered.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7517I: Registering policy type: type=%(1)s,editable=%(2)s,mask=[%(3)s],encrypted=%(4)s,desc=%(5)s**

**Explanation:**

*%(type)* is the type. *%(edit)* is the editable flag. *%(mask)* is the mask. *%(encrypt)* is the encrypt. *%(desc)* is the encrypt. The policy type was registered.

**User Response:**

- No action is needed.

**SOAC7518E: Error registering policy type: %(1)s**

**Explanation:**

*%(msgid)* is the message id. The policy type was not registered.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7519I: Policy type successfully registered.**

**Explanation:**

The policy type was registered successfully.

**User Response:**

- No action is needed.

**SOAC7520I: Running script for service implementation [%(1)s]...**

**Explanation:**

*%(impl)* is the implementation name. The implementation script is running.

**User Response:**

- No action is needed.

**SOAC7521I: Finished script for service implementation [%(1)s].**

**Explanation:**

*%(impl)* is the implementation name. The implementation script completed.

**User Response:**

- No action is needed.

**SOAC7522I: Processed script parameters.**

**Explanation:**

The parameters are processed.

**User Response:**

- No action is needed.

**SOAC7523I: %(1)s=%(2)s**

**Explanation:**

The options are displayed.

**User Response:**

- No action is needed.

**SOAC7524I: Defining new service: service=[%(1)s],parentGroup=[%(2)s],enabled=%(3)s,description=[%(4)s],type=%(5)s**

**Explanation:**

*%(svc)* is the service name. *%(parent)* is the parent name. *%(enabled)* is the enabled name. *%(desc)* is the description name. *%(type)* is the type name. The service is registered.

**User Response:**

- No action is needed.

**SOAC7525I: Service definition successfully created.**

**Explanation:**

The service is registered successfully.

**User Response:**

- No action is needed.

**SOAC7526E: Error creating new service definition: [%(1)s]**

**Explanation:**

An error occurred.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7527I: Defined new operation: service=[%(1)s],operation=%(2)s,enabled=%(3)s,description=[%(4)s]**

**Explanation:**

*%(svc)* is the service. *%(op)* is the operation. *%(enabled)* is the enabled flag. *%(desc)* is the description. The operation is registered.

**User Response:**

- No action is needed.

**SOAC7528I: Operation definition successfully created.**

**Explanation:**

the operation is registered successfully.

**User Response:**

- No action is needed.

**SOAC7529E: Error creating new operation definition: [%(1)s]**

**Explanation:**

An error occurred.

**User Response:**

- Review the configuration data provided to the script.

**SOAC7530E: Error occurred - See the log for details.**

**Explanation:**

A severe error occurred that was not expected.

**User Response:**

- Review the configuration data provided to the script. Study the information provided in the error.

**SOAC7531I: Log: %(1)s**

**Explanation:**

The log file is located at the specified location.

**User Response:**

- Review the log file for additional information about the execution of the task.

**SOAC7532E: Environment variable WAS_HOME is required.**

**Explanation:**

The WAS_HOME environment variable is required.

**User Response:**

- Review the log file for additional information about the execution of the task.

**SOAC7533I: Processing: %(1)s**

**Explanation:**

The processing message gives status information about how the script is progressing.

**User Response:**

- Review the log file for additional information about the execution of the task.

**SOAC7534I: Restarting node agent: %(1)s**

**Explanation:**

*%(node)* is the name of the node on which the node agent is running. This message gives status information when a node agent is restarting.

**User Response:**

- Review the log file for additional information about the execution of the task.

**SOAC7535I: Node agent [%(1)s] is shut down.**

**Explanation:**

*%(node)* is the name of the node on which the node agent is running. This message gives status information when a node agent is shut down.

**User Response:**

- Review the log file for additional information about the execution of the task.

**SOAC7536I: Node agent [%(1)s] is started.**

**Explanation:**

*%(node)* is the name of the node on which the node agent is running. This message gives status information when a node agent is started.

**User Response:**

- Review the log file for additional information about the execution of the task.

**SOAC7537I: WAS Configuration Topology Hint:**

**Explanation:**

This message gives a hint as to the application server topology the could be used in the configuration user interface.

**User Response:**

- This information is provided as a convenience.

**SOAC7538I: Synchronizing node: %(1)s**
**Explanation:**

*%(node)* is the name of the node that is synchronizing with the deployment manager. This message gives status information when a node is synchronizing with the deployment manager.

**User Response:**

- This information is provided as a convenience.

**SOAC7539I: Publishing Common Runtime files to node: %(1)s**
**Explanation:**

*%(node)* is the name of the node that is receiving Common Runtime files. This message gives status information when Common Runtime files are being published to a node.

**User Response:**

- This information is provided as a convenience.

**SOAC7540I: Publishing AG Runtime files to node: %(1)s**
**Explanation:**

*%(node)* is the name of the node that is receiving AG Runtime files. This message gives status information when AG Runtime files are being published to a node.

**User Response:**

- This information is provided as a convenience.

## Messages_en_US

This pseudo class documents the set of messages that can be issued by the IBM Telecom Web Services Server components.

**SOAM0001I: Privacy Check reported the following error: %1.**
**Explaination:**

Privacy Check reported the following error: %1.

**Response:**

Contact the system administrator.

**SOAM0002I: Common Layer reported the following error: %1.**
**Explaination:**

Common Layer reported the following error: %1.

**Response:**

Contact the system administrator.

**SOAM0003I: Admission Control reported the following error: %1.**
**Explaination:**

Admission Control reported the following error: %1.

**Response:**

Contact the system administrator.

**SOAM0004I: No message is defined for fault/alarm %1.**

**Explaination:**

No message is defined for fault/alarm %1.

**Response:**

Contact the system administrator.

**SOAM0005I: Traffic Shaper reported the following error: %1.**

**Explaination:**

Traffic Shaper reported the following error: %1.

**Response:**

Contact the system administrator.

**SOAM0006I: Service %1 was not implemented.**

**Explaination:**

Service %1 was not implemented.

**Response:**

Contact the system administrator.

**SOAM0007I: Asynchronous Service Implementation reported the following error: %1.**

**Explaination:**

Asynchronous Service Implementation reported the following error: %1.

**Response:**

Contact the system administrator.

**SOAM0008I: Asynchronous service message %1 was not enqueued.**

**Explaination:**

Asynchronous service message %1 was not enqueued.

**Response:**

Contact the system administrator.

**SOAM0009I: The backend service implementation encountered an exception: %1.**

**Explaination:**

The backend service implementation encountered an exception: %1.

**Response:**

Contact the system administrator.

**SOAM0010I: Address plan %1 was not available.**

**Explaination:**

Address plan %1 was not available.

**Response:**

Contact the system administrator.

**SOAM0011I: The transaction state transition took too long: %1.**

**Explaination:**

The transaction state transition took too long: %1.

**Response:**

Contact the system administrator.

**SOAM0012I: Service usage record writing reported the following error: %1.**

**Explaination:**

Service usage record writing reported the following error: %1.

**Response:**

Contact the system administrator.

**SOAM0013I: Group resolution is disabled: %1.**

**Explaination:**

Group resolution is disabled: %1.

**Response:**

Contact the system administrator

**SOAM0018E: The request contains incorrect semantic content or is not structured properly.**

**Explaination:**

The request contains incorrect semantic content or is not structured properly.

**Response:**

Check the input values and resend the request.

**SOAM0019E: The Notification Management component could not process this request due to a service error.**

**Explaination:**

The Notification Management component could not process this request due to a service error.

**Response:**

Contact the system administrator.

**SOAM0020E: The Notification Management component has no current or live notification with the specified requester, service and correlator.**

**Explaination:**

The Notification Management component has no current or live notification with the specified requester, service and correlator.

**Response:**

Check the request to ensure that the requester, service, and correlator identify a known notification. Then resend the request.

**SOAM0021E: The specified requester, service, and correlator are already registered with Notification Management.**

**Explaination:**

The specified requester, service, and correlator are already registered with Notification Management.

**Response:**

Check the request to ensure that the requester, service, and correlator are not already registered with Notification Management. Then resend the request.

**SOAM0800EI: An unexpected error occurred %1.**

**Explaination:**

Internal Error: The following unexpected error occurred: %1 .

**Response:**

Contact the system administrator.

**SOAM0801EI: CommonLayer reports null response.**

**Explaination:**

The CommonLayer reports that it has received a null response.

**Response:**

Contact the system administrator.

**SOAM0802EI: Database reports SQL exception %1.**

**Explaination:**

The database reports the following SQL exception: %1.

**Response:**

Contact the system administrator.

**SOAM0803EI: An unexpected database error occurred: %1. The transaction ID is %2.**

**Explaination:**

Internal Error: An unexpected database error occurred: %1. The transaction ID is %2.

**Response:**

Contact the system administrator.

**SOAM0804EI: The address pattern %1 was not valid.**

**Explaination:**

The address pattern %1 was not valid.

**Response:**

Contact the system administrator.

**SOAM0805EI: The Third Party Call Web Service could not read the service policy value for %1. The error was %2.**

**Explaination:**

The Third Party Call Web Service could not read the service policy value for %1. The error was %2.

**Response:**

Contact the system administrator.

**SOAM0806EI: Failed on policy:%1=%2 while field:%3=%4.**

**Explaination:**

Failed on policy:%1=%2 while field:%3=%4.

**Response:**

Contact the system administrator.

**SOAM0807EI: The specified service policy was not valid. The default value will be used.**

**Explaination:**

The specified service policy was not valid. The default value will be used.

**Response:**

Contact the system administrator.

**SOAM0808EI: The Third Party Call Web service could not read the management bean value for %1. The error was %2.**

**Explaination:**

The Third Party Call Web service could not read the management bean value for %1. The error was %2.

**Response:**

Contact the system administrator.

**SOAM0809EI: Failed on MBean attribute:%1=%2 while field:%3=%4.**

**Explaination:**

Failed on MBean attribute:%1=%2 while field:%3=%4.

**Response:**

Contact the system administrator.

**SOAM0810EI: Internal Error: An attempt was made to get a service usage record for an unknown operation (%1).**

**Explaination:**

Internal Error: An attempt was made to get a service usage record for an unknown operation (%1).

**Response:**

Contact the system administrator.

**SOAM0811EI: Could not clean up a %1 resource: %2.**

**Explaination:**

Could not clean up a %1 resource: %2.

**Response:**

Contact the system administrator.

**SOAM0812EI:The JNDI Name was not resolved %1.**

**Explaination:**

Internal Error: The JNDI Name was not resolved %1.

**Response:**

Contact the system administrator

**SOAM0900EI: The service implementation was not found.**

**Explaination:**

The service implementation was not found

**Response:**

Contact the system administrator.

**SOAM0901EI: A server internal error occurred.**

**Explaination:**

A server internal error occurred.

**Response:**

Contact the system administrator.

**SOAM0902EI: A database failure occurred.**

**Explaination:**

A database failure occurred.

**Response:**

Contact the system administrator.

**SOAM0903EI: Could not discover MPCC SCF using the specified name.**

**Explaination:**

Could not discover MPCC SCF using the specified name.

**Response:**

Contact the system administrator.

**SOAM0904EI: The operation %1 is not supported by the Parlay gateway in its current configuration.**

**Explaination:**

The operation %1 is not supported by the Parlay gateway in its current configuration.

**Response:**

Contact the system administrator.

**SOAM0905EI: An error occurred while invoking an EJB operation: %1.**

**Explaination:**

An error occurred while invoking an EJB operation: %1.

**Response:**

Contact the system administrator.

**SOAM1503E: The delivery status data records could not be retrieved for the following reason: %1.**

**Explaination:**

The delivery status data records could not be retrieved for the following reason %1.

**Response:**

Contact the system administrator.

**SOAM1506E: The correlator provided is in use: %1.**

**Explaination:**

The correlator provided is in use: %1.

**Response:**

Select a correlator that is not already in use, and retry.

**SOAM1508E: The correlator provided is in use: %1.**

**Explaination:**

The correlator provided is in use: %1.

**Response:**

Select a correlator that is not already in use, and retry.

**SOAM1509E: The criteria provided are in use: %1.**
**Explaination:**

The criteria provided are in use: %1.

**Response:**

Select criteria that are not already in use, and retry.

**SOAM1520I: Null input value for the addresses parameter.**
**Explaination:**

No value was specified for the addresses parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1521I: Empty string input value for an address parameter.**
**Explaination:**

An empty string was specified for one of the address parameters. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1522I: Null input value for the message parameter.**
**Explaination:**

No value was specified for the message parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1523I: Empty string input value for the message parameter.**
**Explaination:**

An empty string was specified for the message parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1525E: Creating Send SMS data record failed.**
**Explaination:**

An attempt to create a Send SMS data record was not successful.

**Response:**

Contact the system administrator.

**SOAM1526I: Null input value for the requestIdentifier parameter.**
**Explaination:**

No value was specified for the requestIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1527I: Empty string input value for the requestIdentifier parameter.**
**Explaination:**

An empty string was specified for the requestIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1528I: Could not find delivery status for this request identifier.**
**Explaination:**

Could not find delivery status for this request identifier.

**Response:**

Check that you have specified a valid value for the request identifier, and retry.

**SOAM1529E: Unknown delivery status value.**
**Explaination:**

Unknown delivery status value.

**Response:**

Contact the system administrator.

**SOAM1530E: Requester ID is null in MDS context.**
**Explaination:**

Requester ID is null in MDS context.

**Response:**

Contact the system administrator.

**SOAM1531E: MDS context is null.**
**Explaination:**

MDS context is null.

**Response:**

Contact the system administrator.

**SOAM1532E: Error when creating notification record for SendSms request.**
**Explaination:**

Error when creating notification record for SendSms request.

**Response:**

Contact the system administrator.

**SOAM1533E: Problem obtaining Parlay Connector.**
**Explaination:**

Problem obtaining Parlay Connector.

**Response:**

Make sure the Parlay Connector is started.

**SOAM1534I: Privacy check failed for this address.**
**Explaination:**

Privacy check failed for this address.

**Response:**

Check to make sure that privacy is configured correctly for this address.

**SOAM1535I: Empty string input value for correlator parameter.**

**Explaination:**

An empty string was specified for the correlator parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1536I: Null input value for correlator parameter.**

**Explaination:**

No value was specified for the correlator parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1537I: Null input value for reference parameter.**

**Explaination:**

No value was specified for the correlator parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1538I: Null input value for endpoint parameter.**

**Explaination:**

No value was specified for the endpoint parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1539I: Empty string input value for endpoint parameter.**

**Explaination:**

An empty string was specified for the endpoint parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1540I: Null input value for interfaceName parameter.**

**Explaination:**

No value was specified for the interfaceName parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1541I: Empty string input value for interfaceName parameter.**

**Explaination:**

An empty string was specified for the interfaceName parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1542I: Null input value for smsServiceActivationNumber parameter.**

**Explaination:**

No value was specified for the smsServiceActivationNumber parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1543I: Empty string input value for smsServiceActivationNumber parameter.**

**Explaination:**

An empty string was specified for the smsServiceActivationNumber parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1544I: Null input value for registrationIdentifier parameter.**

**Explaination:**

No value was specified for the registrationIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1545I: Empty string input value for registrationIdentifier parameter.**

**Explaination:**

An empty string was specified for the registrationIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1555E: Unexpected operation for this method.**

**Explaination:**

This method is not expected to be called by the operation being performed.

**Response:**

Contact the system administrator.

**SOAM1556E: URISyntaxException during creation of URI.**

**Explaination:**

A URISyntaxException occurred during creation of the URI.

**Response:**

Contact the system administrator.

**SOAM1557E: Error creating notification record for startNotification call.**

**Explaination:**

An error occurred during creation of the notification record for the startNotification call.

**Response:**

Contact the system administrator.

**SOAM1558I: The SMS service activation number was not valid.**

**Explaination:**

The SMS service activation number was not valid.

**Response:**

Enter a valid value for the SMS service activation number, and retry.

**SOAM1559I: The SMS service is disabled.**

**Explaination:**

The SMS service is disabled.

**Response:**

Enable the SMS service if you want to use it.

**SOAM1560W: Charging is not supported.**

**Explaination:**

Charging is not supported.

**Response:**

Change the service policy setting if you want to support charging.

**SOAM1561W: The registration identifier does not exist.**

**Explaination:**

The registration identifier does not exist.

**Response:**

Check that you have specified a valid value for the registration identifier, and retry.

**SOAM1562I: Null input value for the image parameter.**

**Explaination:**

No value was specified for the image parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1563I: Length 0 for image parameter.**

**Explaination:**

A string of zero length was specified for the image parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1564I: Null input value for the ringtone parameter.**

**Explaination:**

No value was specified for the ringtone parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1565I: Empty string input value for ringtone parameter.**

**Explaination:**

An empty string was specified for the ringtone parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1566W: Problem in UDH Encoding.**

**Explaination:**

Problem in UDH Encoding.

**Response:**

Check that you have specified a valid value for UDH Encoding, and retry.

**SOAM1567E: The service policy %1 is not in the MdsContext.**

**Explaination:**

The service policy %1 is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM1568E: The service policy value is not as expected. Service Policy: %1. Policy Value: %2**

**Explaination:**

The service policy value is not as expected. Service Policy: %1. Policy Value: %2

**Response:**

Contact the system administrator.

**SOAM1569I: Null input value for the smsFormat parameter.**

**Explaination:**

No value was specified for the SmsFormat parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM1575E: Could not find the notification data for the correlator.**

**Explaination:**

Could not find the notification data for the correlator.

**Response:**

Contact the system administrator.

**SOAM1576E: There is no connection to the database.**

**Explaination:**

There is no connection to the database.

**Response:**

Contact the system administrator.

**SOAM1577E: An unknown error occurred.**

**Explaination:**

An unknown error occurred. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1578E: Could not create the record of the session object.**

**Explaination:**

Could not create the record of the session object. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1579E: A problem occurred while updating the status for the destination.**

**Explaination:**

A problem occurred while updating the status for the destination. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1580E: A problem occurred while finding the segment status for the transaction.**

**Explaination:**

A problem occurred while finding the segment status for the transaction. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1581E: Could not find the notification data for the assignment ID.**

**Explaination:**

Could not find the notification data for the assignment ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1582E: A problem occurred while finding the first word of the message.**

**Explaination:**

A problem occurred while finding the first word of the message. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1583E: A problem occurred with a key when storing the receive data.**

**Explaination:**

A problem occurred with a key when storing the receive data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1584E: A problem occurred with a create operation when storing the receive data.**

**Explaination:**

A problem occurred with a create operation when storing the receive data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1585E: Could not find the notification data for this send transaction ID.**

**Explaination:**

Could not find the notification data for this send transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1586E: Could not perform the notification for this transaction ID.**

**Explaination:**

Could not perform the notification for this transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1587E: Could not find the notification data for this receive transaction ID.**

**Explaination:**

Could not find the notification data for this receive transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1588E: Could not find the notification data to delete.**

**Explaination:**

Could not find the notification data to delete. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1589E: Could not remove the session object for the assignment ID.**

**Explaination:**

Could not remove the session object for the assignment ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1590E: Could not find expired send data.**

**Explaination:**

Could not find expired send data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1591E: Could not remove expired send data.**

**Explaination:**

Could not remove expired send data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1592E: Could not find expired notification data.**

**Explaination:**

Could not find expired notification data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1593E: Could not remove expired notification data.**

**Explaination:**

Could not remove expired notification data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1594E: Could not find expired received data.**

**Explaination:**

Could not find expired received data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1595E: Could not remove expired received data.**

**Explaination:**

Could not remove expired received data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1596E: Could not find expired session object data.**

**Explaination:**

Could not find expired session object data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1597E: Could not remove expired session object data.**

**Explaination:**

Could not remove expired session object data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1598E: Could not reach a client to notify it of a Notification event.**

**Explaination:**

Could not reach a client to notify it of a Notification event.

**Response:**

Check the endpoint address and the PxNotification configuration.

**SOAM1599E: Could not access a table: the table is not defined, or the initial data source was not created.**

**Explaination:**

Could not access a table: the table is not defined, or the initial data source was not created.

**Response:**

Make sure that the table you are trying to access exists, and check the configuration for the data source.

**SOAM1600E: The database service is not running, or database authentication failed.** **Explaination:**

The database service is not running, or database authentication failed.

**Response:**

Start the database service or check the authentication credentials. Then retry.

**SOAM1650E: The service policy service.config.target.Aliases is not in the MdsContext.**

**Explaination:**

The service policy service.config.target.Aliases is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM1651E: The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.**

**Explaination:**

The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM1652E: No alias names were defined in the Administration Console.**

**Explaination:**

No alias names were defined in the Administration Console.

**Response:**

Contact the system administrator.

**SOAM1653E: An MBeanException occurred while attempting to get the MBean attribute.**

**Explaination:**

An MBeanException occurred while attempting to get the MBean attribute.

**Response:**

Contact the system administrator.

**SOAM1654E: could not connect to the specified server.**

**Explaination:**

Could not connect to the specified server. A ConnectorException was encountered.

**Response:**

Contact the system administrator

**SOAM1655E: An internal error occurred in the SMPP Session Manager.**

**Explaination:**

An internal error occurred in the SMPP Session Manager (SmppSessionManager).

**Response:**

Contact the system administrator.

**SOAM1656E: An error occurred while sending the message to the SMSC.**

**Explaination:**

An error occurred while sending the message to the SMSC.

**Response:**

Contact the system administrator.

**SOAM1657E: The Enquire Link Response message was not sent.**

**Explaination:**

The Enquire Link Response message was not sent.

**Response:**

Contact the system administrator.

**SOAM1658E: The Deliver SM Response message was not sent.**

**Explaination:**

The Deliver SM Response message was not sent.

**Response:**

Contact the system administrator.

**SOAM1659E: The Generic Non-Acknowledge message was not sent.**

**Explaination:**

The Generic Non-Acknowledge message was not sent.

**Response:**

Contact the system administrator.

**SOAM1660E: No alias name was defined in the Administration Console. %1**

**Explaination:**

No alias name was defined in the administration console. %1

**Response:**

Contact the system administrator.

**SOAM1663E: The specified endpoint URI for Notification was not valid.**

**Explaination:**

The specified endpoint URI for Notification was not valid.

**Response:**

Enter the correct URI and retry.

**SOAM1664E: Null input value for the charging description.**

**Explaination:**

No value was specified for the charging description. A value must be specified.

**Response:**

Enter a valid charging description and retry.

**SOAM1665E: Empty string input value for the charging description.**

**Explaination:**

An empty string was specified for the charging description. A value must be specified.

**Response:**

Enter a valid charging description and retry.

**SOAM1800E: The service policy service.standard.VirtualNumber is not in the MdsContext.**

**Explaination:**

The service policy service.standard.VirtualNumber is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM1801E: The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.**

**Explaination:**

The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.

**Response:**

Contact the system administrator

**SOAM1802E: An AttributeNotFoundException occurred while attempting to get the MBean attribute.**

**Explaination:**

An AttributeNotFoundException occurred while attempting to get the MBean attribute.

**Response:**

Contact the system administrator.

**SOAM1803E: An MBeanException occurred while attempting to get the MBean attribute.**

**Explaination:**

An MBeanException occurred while attempting to get the MBean attribute.

**Response:**

Contact the system administrator.

**SOAM1804E: A ReflectionException occurred while attempting to get the MBean attribute.**

**Explaination:**

A ReflectionException occurred while attempting to get the MBean attribute.

**Response:**

Contact the system administrator.

**SOAM1805E: An openMultiMediaMessaging Parlay call failed with an exception.**

**Explaination:**

An openMultiMediaMessaging Parlay call failed with an exception.

**Response:**

Contact the system administrator.

**SOAM1806E: An openMultiMediaMessaging Parlay call failed with a P_INVALID_ADDRESS exception.**

**Explaination:**

An openMultiMediaMessaging Parlay call failed with a P_INVALID_ADDRESS exception.

**Response:**

Contact the system administrator.

**SOAM1807E: close call for IpMultiMediaMessaging reference failed.**

**Explaination:**

close call for IpMultiMediaMessaging reference failed.

**Response:**

Contact the system administrator.

**SOAM1808E: A createNotification call for IpMultiMediaMessagingManager failed.**

**Explaination:**

A createNotification call for IpMultiMediaMessagingManager failed.

**Response:**

Contact the system administrator.

**SOAM1809E: A destroyNotification call for IpMultiMediaMessagingManager failed.**

**Explaination:**

A destroyNotification call for IpMultiMediaMessagingManager failed.

**Response:**

Contact the system administrator.

**SOAM1810E: A sendMessageReq call for IpMultiMediaMessaging failed.**

**Explaination:**

A sendMessageReq call for IpMultiMediaMessaging failed.

**Response:**

Contact the system administrator.

**SOAM1811E: A setConfirmDelivery call for MmmCallBack failed.**

**Explaination:**

A setConfirmDelivery call for MmmCallBack failed.

**Response:**

Contact the system administrator.

**SOAM1812E: Could not get EJB reference for MmmCallBack.**

**Explaination:**

Could not get EJB reference for MmmCallBack.

**Response:**

Contact the system administrator.

**SOAM2003E: The delivery status data records could not be retrieved for the following reason: %1.**

**Explaination:**

The delivery status data records could not be retrieved for the following reason: %1.

**Response:**

Contact the system administrator.

**SOAM2006E: The correlator provided is in use: %1.**
**Explaination:**

The correlator provided is in use: %1.

**Response:**

Enter a valid value and retry.

**SOAM2008E: The correlator provided is in use: %1.**
**Explaination:**

The correlator provided is in use: %1.

**Response:**

Enter a valid value and retry.

**SOAM2009E: The criteria provided is in use: %1.**
**Explaination:**

The criteria provided is in use: %1.

**Response:**

Enter a valid value and retry.

**SOAM2020I: Null input value for the addresses parameter.**
**Explaination:**

No value was specified for the addresses parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2021I: Empty string input value for one of the address parameters.**
**Explaination:**

An empty string was specified for one of the address parameters. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2025E: Creating Send Message data record failed.**
**Explaination:**

Creating Send Message data record failed.

**Response:**

Contact the system administrator.

**SOAM2026I: Null input value for requestIdentifier parameter.**
**Explaination:**

No value was specified for the requestIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2027I: Empty string input value for requestIdentifier parameter.**

**Explaination:**

An empty string was specified for the requestIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2028I: Could not find the delivery status for this request identifier.**

**Explaination:**

Could not find the delivery status for this request identifier.

**Response:**

Enter a valid value and retry.

**SOAM2029E: Unknown delivery status value.**

**Explaination:**

The value specified for delivery status was unknown.

**Response:**

Contact the system administrator.

**SOAM2030E: The requester ID is null in the MDS context.**

**Explaination:**

The requester ID is null in the MDS context.

**Response:**

Contact the system administrator.

**SOAM2031E: The MDS context is null.**

**Explaination:**

The MDS context is null.

**Response:**

Contact the system administrator.

**SOAM2032E: An error occurred while creating notification record for SendMessage request.**

**Explaination:**

An error occurred while creating notification record for SendMessage request.

**Response:**

Contact the system administrator.

**SOAM2035I: Empty string input value for correlator parameter.**

**Explaination:**

An empty string was specified for the correlator parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2036I: Null input value for correlator parameter.**

**Explaination:**

No value was specified for the correlator parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2037I: Null input value for reference parameter.**

**Explaination:**

No value was specified for the reference parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2038I: Null input value for endpoint parameter.**

**Explaination:**

No value was specified for the endpoint parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2039I: Empty string input value for endpoint parameter.**

**Explaination:**

An empty string was specified for the endpoint parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2040I: Null input value for interfaceName parameter.**

**Explaination:**

No value was specified for the interfaceName parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2041I: Empty string input value for interfaceName parameter.**

**Explaination:**

An empty string was specified for the interfaceName parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2042I: Null input value for messageServiceActivationNumber parameter.**

**Explaination:**

No value was specified for the messageServiceActivationNumber parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2043I: Empty string input value for messageServiceActivationNumber parameter.**

**Explaination:**

An empty string was specified for the messageServiceActivationNumber parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2044I: Null input value for registrationIdentifier parameter.**

**Explaination:**

No value was specified for the registrationIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2045I: Empty string input value for registrationIdentifier parameter.**

**Explaination:**

An empty string was specified for the registrationIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2046I: Null input value for messageRefIdentifier parameter.**

**Explaination:**

No value was specified for the messageRefIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2047I: Empty string input value for messageRefIdentifier parameter.**

**Explaination:**

An empty string was specified for the messageRefIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM2048I: The type value specified for messageRefIdentifier was not valid.**

**Explaination:**

The type value specified for messageRefIdentifier was not valid.

**Response:**

Enter a valid value and retry.

**SOAM2055E: Unexpected operation for this method.**

**Explaination:**

This method is not expected to be called by this operation.

**Response:**

Contact the system administrator

**SOAM2057E: An error occurred creating a notification record for the startNotification call.**

**Explaination:**

An error occurred while creating a notification record for the startNotification call.

**Response:**

Contact the system administrator

**SOAM2058I: The Message Service activation number was not valid.**

**Explaination:**

The Message service activation number was not valid.

**Response:**

Enter valid value for message service activation number.

**SOAM2059I: The MMS service is disabled.**

**Explaination:**

The MMS service is disabled.

**Response:**

Enable the MMS service and retry.

**SOAM2060W: Charging is not supported.**

**Explaination:**

Charging is not supported.

**Response:**

Change the service policy setting if you want to support charging.

**SOAM2061W: The value specified for registration identifier does not exist.**

**Explaination:**

The value specified for registration identifier does not exist.

**Response:**

Enter a valid value and retry.

**SOAM2062W: The message with this MessageRef Identifier has expired.**

**Explaination:**

The message with this MessageRef Identifier has expired.

**Response:**

Enter a valid value and retry.

**SOAM2063W: The value specified for MessageRef Identifier does not exist.**

**Explaination:**

The value specified for MessageRef Identifier does not exist.

**Response:**

Enter a valid value and retry.

**SOAM2064W: The message with this MessageRef Identifier is new and has not been retrieved.**

**Explaination:**

The message with this MessageRef Identifier is new and has not been retrieved.

**Response:**

Retrieve new messages, and retry.

**SOAM2069E: Could not find an attachment ID corresponding to this transaction ID.**

**Explaination:**

Could not find an attachment ID corresponding to this transaction ID.

**Response:**

Contact the system administrator.

**SOAM2070E: Could not find attachment data corresponding to this attachment ID.**

**Explaination:**

Could not find attachment data corresponding to this attachment ID.

**Response:**

Contact the system administrator.

**SOAM2071E: The attachments were not deleted. Could not find an attachment ID corresponding to this transaction ID.**

**Explaination:**

The attachments were not deleted. Could not find an attachment ID corresponding to this transaction ID.

**Response:**

Contact the system administrator.

**SOAM2072E: The attachments were not deleted. Could not find attachment data corresponding to this attachment ID.**

**Explaination:**

The attachments were not deleted. Could not find attachment data corresponding to this attachment ID.

**Response:**

Contact the system administrator.

**SOAM2073E: A problem occurred while deleting attachment content.**

**Explaination:**

A problem occurred while deleting attachment content.

**Response:**

Contact the system administrator.

**SOAM2074E A problem occurred while deleting attachment metadata.**

**Explaination:**

A problem occurred while deleting attachment metadata.

**Response:**

Contact the system administrator.

**SOAM2075E: Could not find the notification data for the correlator.**

**Explaination:**

Could not find the notification data for the correlator.

**Response:**

Contact the system administrator.

**SOAM2076E: There is no connection to the database.**

**Explaination:**

There is no connection to the database.

**Response:**

Contact the system administrator.

**SOAM2077E: An unknown error occurred.**

**Explaination:**

An unknown error occurred. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2079E: A problem occurred while updating the status for the destination.**

**Explaination:**

A problem occurred while updating the status for the destination. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2080E: A duplicate key problem occurred while storing attachment content.**

**Explaination:**

A duplicate key problem occurred while storing attachment content. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2081E: A problem occurred while storing attachment content.**

**Explaination:**

A problem occurred while storing attachment content. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2082E: A problem occurred while storing attachment metadata.**

**Explaination:**

A problem occurred while storing attachment metadata. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2083E: A key-related problem occurred while storing the receive data.**

**Explaination:**

A key-related problem occurred while storing the receive data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2084E: A create problem occurred while storing the receive data.**

**Explaination:**

A create problem occurred while storing the receive data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2085E: Could not find the notification data for this send transaction ID.**

**Explaination:**

Could not find the notification data for this send transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2086E: Could not perform the notification for this transaction ID.**

**Explaination:**

Could not perform the notification for this transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2088E: Could not find the notification data to delete.**

**Explaination:**

Could not find the notification data to delete. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2090E: Could not find expired send data.**

**Explaination:**

Could not find expired send data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2091E: Could not remove expired send data.**

**Explaination:**

Could not remove expired send data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2092E: Could not find expired notification data.**

**Explaination:**

Could not find expired notification data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2093E: Could not remove expired notification data.**

**Explaination:**

Could not remove expired notification data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2094E: Could not find expired received data.**
**Explaination:**

Could not find expired received data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2095E: Could not remove expired received data.**
**Explaination:**

Could not remove expired received data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2096E: Could not find expired session object data.**
**Explaination:**

Could not find expired session object data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2097E: Could not remove expired session object data. Check the error log.**
**Explaination:**

Could not remove expired session object data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM2098E: Could not reach a client to notify it of a Notification event.**
**Explaination:**

Could not reach a client to notify it of a Notification event.

**Response:**

Check the endpoint address and the PxNotification configuration, and retry.

**SOAM2099E: The table is not defined, or the data source could not be created.**
**Explaination:**

The table you are trying to access is not defined, or a problem occurred during initial creation of the data source.

**Response:**

Make sure that the table exists, or check the data source configuration.

**SOAM2148E: The service policy %1 is not in the MdsContext.**
**Explaination:**

The service policy %1 is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM2149E: The service policy value is not as expected. Service Policy: %1. Policy Value: %2**

**Explaination:**

The service policy value is not as expected. Service Policy: %1. Policy Value: %2

**Response:**

Contact the system administrator.

**SOAM2150E: The service policy service.common.target.Aliases is not in the MdsContext.**

**Explaination:**

The service policy service.common.target.Aliases is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM2151E: The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.**

**Explaination:**

The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM2152E: No alias names were defined in the Administration Console.**

**Explaination:**

No alias names were defined in the Administration Console.

**Response:**

Contact the system administrator.

**SOAM2153E: An MBeanException occurred while attempting to get the MBean attribute.**

**Explaination:**

An MBeanException occurred while attempting to get the MBean attribute.

**Response:**

Contact the system administrator.

**SOAM2154E: The MM7 JCA connector was found.**

**Explaination:**

A ConnectorException occurred. The MM7 JCA connector was not found.

**Response:**

Contact the system administrator.

**SOAM2156E: An error occurred while sending the message to the MMSC.**

**Explaination:**

An error occurred while sending the message to the MMSC.

**Response:**

Contact the system administrator.

**SOAM2160E: No alias name was defined in the Administration Console. %1**

**Explaination:**

No alias name was defined in the Administration Console. %1

**Response:**

Contact the system administrator.

**SOAM2163E: The specified endpoint URI for Notification was not valid.**

**Explaination:**

The specified endpoint URI for Notification was not valid.

**Response:**

Enter the correct URI and retry.

**SOAM3800E: An unexpected internal error occurred within the Terminal Status service.**

**Explaination:**

An unexpected internal error occurred within the Terminal Status service.

**Response:**

Contact the system administrator.

**SOAM3808W: An internal error occurred. An attempt was made to get a service usage record for an unknown Terminal Status operation.**

**Explaination:**

An internal error occurred. An attempt was made to get a service usage record for an unknown Terminal Status operation.

**Response:**

Contact the system administrator.

**SOAM3809E: An unexpected database error occurred in the Terminal Status notification service.**

**Explaination:**

An unexpected database error occurred in the Terminal Status notification service.

**Response:**

Retry the request. If the problem persists, contact the system administrator.

**SOAM3810E: Could not create a database entry for a Terminal Status notification service request.**

**Explaination:**

Could not create a database entry for a Terminal Status notification service request.

**Response:**

Retry the request. If the problem persists, contact the system administrator.

**SOAM3811E: Could not locate a database entry for a Terminal Status notification service request.**

**Explaination:**

Could not locate a database entry for a Terminal Status notification service request.

**Response:**

Retry the request. If the problem persists, contact the system administrator.

**SOAM3812W: Could not reach a client to notify it of a Terminal Status notification event.**

**Explaination:**

Could not reach a client to notify it of a Terminal Status notification event.

**Response:**

Check the endpoint address and the PxNotification configuration, and retry.

**SOAM3813W: Could not clean up a Terminal Status notification resource. An error occurred.**

**Explaination:**

Could not clean up a Terminal Status notification resource. An error occurred.

**Response:**

Retry the request. If the problem persists, contact the system administrator.

**SOAM3814E: An internal error occurred while attempting to resolve a Terminal Status JNDI reference.**

**Explaination:**

An internal error occurred while attempting to resolve a Terminal Status JNDI reference.

**Response:**

Contact the system administrator.

**SOAM3815I: The Terminal Status service could not read a service policy value.**

**Explaination:**

The Terminal Status service could not read a service policy value.

**Response:**

Make sure a valid value is specified for the service policy, and retry.

**SOAM3816I: The Terminal Status service could not read a management bean value.**

**Explaination:**

The Terminal Status service could not read a management bean value.

**Response:**

Make sure a valid value is specified for the management bean, and retry.

**SOAM3900E: A Terminal status service request failed. A Parlay error occurred.**

**Explaination:**

A Terminal Status service request failed. A Parlay error occurred.

**Response:**

Check the Parlay configuration and the Parlay gateway, and retry.

**SOAM3902E: A Terminal Status service request failed. The synchronization object timed out.**

**Explaination:**

A Terminal Status service request failed. The synchronization object timed out.

**Response:**

Verify that the Parlay gateway is operational, or increase the timeout value. Then retry.

**SOAM3904E: A Terminal Status service request failed. A Parlay Unauthorized network error occurred.**

**Explaination:**

A Terminal Status service request failed. A Parlay Unauthorized network error occurred.

**Response:**

Check the Parlay configuration and retry.

**SOAM3905E: A Terminal Status service request failed. A Parlay Position method failure occurred.**

**Explaination:**

A Terminal Status service request failed. A Parlay Position method failure occurred.

**Response:**

Check the Parlay configuration and retry.

**SOAM3908E: A Terminal Status service request failed. The request could not obtain a compatible service manager object from Parlay.**

**Explaination:**

A Terminal Status service request failed. The request could not obtain a compatible service manager object from Parlay.

**Response:**

Check the Terminal Status service name or the Parlay configuration, then retry.

**SOAM3909E: A Terminal Status service request failed. The operation invoked is not supported.**

**Explaination:**

A Terminal Status service request failed. The operation invoked is not supported.

**Response:**

Make sure the Terminal Status Web service settings match the capabilities of the Parlay gateway. Then retry.

**SOAM3911E: The Terminal Status notification Request failed. The specified correlator is already in use.**

**Explaination:**

The Terminal Status notification Request failed. The specified correlator is already in use.

**Response:**

Provide a unique correlator for this request, and retry.

**SOAM3912E: Could not create a database entry for a Terminal Status notification service request.**

**Explaination:**

Could not create a database entry for a Terminal Status notification service request.

**Response:**

Retry the request. If the problem persists, contact the system administrator.

**SOAM3916I: The Terminal Status notification request failed. The specified correlator is not registered.**

**Explaination:**

The Terminal Status notification request failed. The specified correlator is not registered.

**Response:**

Provide a registered correlator for this request, and retry.

**SOAM3917E: The Terminal Status notification request failed. The specified correlator was not valid.**

**Explaination:**

The Terminal Status notification request failed. The specified correlator was not valid.

**Response:**

Provide a valid correlator for this request, and retry.

**SOAM3918E: A Terminal Status service request failed. The Parlay gateway refused a parameter in the request.**

**Explaination:**

A Terminal Status service request failed. The Parlay gateway refused a parameter in the request.

**Response:**

Check that all of the parameters in the request are valid, and retry.

**SOAM3919E: A Terminal Status service request failed. The Parlay gateway reported a system failure.**

**Explaination:**

A Terminal Status service request failed. The Parlay gateway reported a system failure.

**Response:**

Check the Parlay gateway for error details.

**SOAM3920E: A Terminal Status service request failed. A Parlay Unauthorized Application error occurred.**

**Explaination:**

A Terminal Status service request failed. A Parlay Unauthorized Application error occurred.

**Response:**

Check the Parlay configuration and retry.

**SOAM3921E: A Terminal Status service request failed. An unknown Parlay error occurred.**

**Explaination:**

A Terminal Status service request failed. An unknown Parlay error occurred.

**Response:**

Check the Parlay configuration and the Parlay gateway, then retry.

**SOAM3922E: A Terminal Status notification service received a notification from the Parlay gateway that was not valid. The notification was ignored.**

**Explaination:**

A Terminal Status notification service received a notification from the Parlay gateway that was not valid. The notification was ignored.

**Response:**

Check the Parlay configuration and the Parlay gateway.

**SOAM4000E: An unexpected internal error occurred with the Terminal Location service.**

**Explaination:**

An unexpected internal error occurred with the Terminal Location service.

**Response:**

Contact the system administrator.

**SOAM4008W: An unknown error occurred within the Terminal Location service while attempting to get a service usage record.**

**Explaination:**

An unknown error occurred within the Terminal Location service while attempting to get a service usage record.

**Response:**

Contact the system administrator.

**SOAM4009E: An unexpected database error occurred in the Terminal Location Notification service.**

**Explaination:**

An unexpected database error occurred in the Terminal Location Notification service.

**Response:**

Retry the request. Contact the system administrator if the problem persists.

**SOAM4010E: A database entry could not be created for a Terminal Location Notification service request.**

**Explaination:**

A database entry could not be created for a Terminal Location Notification service request.

**Response:**

Retry the request. Contact the system administrator if the problem persists.

**SOAM4011E: A database entry could not be located for a Terminal Location Notification service request.**

**Explaination:**

A database entry could not be located for a Terminal Location Notification service request.

**Response:**

Retry the request. Contact the system administrator if the problem persists.

**SOAM4012W: Could not reach a client to notify it of a Location Notification event.** **Explaination:**

Could not reach a client to notify it of a Location Notification event.

**Response:**

Check the endpoint address and the PxNotification configuration, then retry.

**SOAM4013W: Could not clean up a Terminal Location Notification resource because an error occurred.**
**Explaination:**

Could not clean up a Terminal Location Notification resource because an error occurred.

**Response:**

Retry the request. Contact the system administrator if the problem persists.

**SOAM4014E: An internal error occurred while attempting to resolve a Terminal Location JNDI reference.**
**Explaination:**

An internal error occurred while attempting to resolve a Terminal Location JNDI reference.

**Response:**

Contact the system administrator.

**SOAM4015I: The Terminal Location service could not read a service policy value.** **Explaination:**

The Terminal Location service could not read a service policy value.

**Response:**

Make sure a valid value is specified for the service policy, and retry.

**SOAM4016I: The Terminal Location service could not read a management bean value.** **Explaination:**

The Terminal Location service could not read a management bean value.

**Response:**

Make sure a valid value is specified for the management bean, and retry.

**SOAM4017I: The service policy value was not valid. The default value will be used.** **Explaination:**

The service policy value was not valid. The default value will be used.

**Response:**

No action is required.

**SOAM4150E: The Terminal Location request failed. A system failure occurred in the in MLP location server.**
**Explaination:**

The Terminal Location request failed. A system failure occurred in the MLP location server.

**Response:**

Contact the system administrator.

**SOAM4151E: The Terminal Location Web service had an authorization problem.**

**Explaination:**

The Terminal Location Web service reported the following error: MLP Unauthorized Network.

**Response:**

Check the configuration for the MLP connector, and retry the operation.

**SOAM4152E: The request failed. An Unknown Subscriber error was reported.**

**Explaination:**

The request failed. An Unknown Subscriber error was reported.

**Response:**

Contact the system administrator.

**SOAM4153E: The request failed. An Absent Subscriber error was reported.**

**Explaination:**

The request failed. An Absent Subscriber error was reported.

**Response:**

Retry at another time.

**SOAM4154E: The request failed. There was congestion in the MLP location server.**

**Explaination:**

The request failed. There was congestion in the MLP location server.

**Response:**

Retry when the location server is less busy.

**SOAM4155E: The request failed. There was congestion in the mobile network.**

**Explaination:**

The request failed. There was congestion in the mobile network.

**Response:**

Retry when there is less traffic in the network.

**SOAM4156E: The response from the MLP location server was corrupted or not valid.**

**Explaination:**

The response from the MLP location server was corrupted or not valid.

**Response:**

Contact the system administrator.

**SOAM4157E: The request failed. There were too many position items.**

**Explaination:**

The request failed for the following reason: Too many position items.

**Response:**

Check your request to ensure it contains the proper number of items, and retry.

**SOAM4158E: The request was rejected. The operation is not supported by the current configuration.**

**Explaination:**

The request was rejected. The operation is not supported by the current configuration.

**Response:**

Make sure that support for this operation is enabled. Then retry.

**SOAM4159E: The request failed. The operation is not allowed by Local regulations.**

**Explaination:**

The request failed for the following reason: Disallowed by local regulations.

**Response:**

Contact the system administrator.

**SOAM4160E: The request failed. The location server is not configured properly.**

**Explaination:**

The request failed for the following reason: Misconfiguration of location server.

**Response:**

Check the configuration of the location server, and retry.

**SOAM4161E: The correlator provided in a start notification request is already registered.**

**Explaination:**

The correlator provided in a start notification request is already registered.

**Response:**

Provide a unique correlator for this request, and retry.

**SOAM4162E: The correlator provided in a end notification request is not registered.**

**Explaination:**

The correlator provided in a end notification request is not registered.

**Response:**

Provide a registered correlator for this request, and retry.

**SOAM4163E: The request failed. It did not contain valid target addresses.**

**Explaination:**

The request failed. It did not contain valid target addresses.

**Response:**

Provide valid target information for the request, and retry.

**SOAM4174E: The correlator provided was not valid.**

**Explaination:**

The correlator provided was not valid.

**Response:**

Provide a valid correlator for the request, and retry.

**SOAM4178E: The Terminal Location Web service reported the following error: MLP Position Method Failure.**

**Explaination:**

The Terminal Location Web service reported the following error: MLP Position Method Failure.

**Response:**

Contact the system administrator.

**SOAM4179E: The Terminal Location Web service reported the following error: Unsupported Protocol Version.**

**Explaination:**

The Terminal Location Web service reported the following error: Unsupported Protocol Version.

**Response:**

Contact the system administrator.

**SOAM4180E: The Terminal Location Web Service reported the following error: Unknown MLP Failure.**

**Explaination:**

The Terminal Location Web Service reported the following error: Unknown MLP Failure.

**Response:**

Contact the system administrator.

**SOAM4181E: The request failed. It contained an incorrect parameter.**

**Explaination:**

The request failed. The MLP location server rejected a request that contained an incorrect parameter.

**Response:**

Contact the system administrator.

**SOAM4182E: The request failed. Positioning is not allowed for the target.**

**Explaination:**

The request failed. The MLP location server reports that positioning is not allowed for the target.

**Response:**

Contact the system administrator.

**SOAM4183E: The MLP location server reported the following error: The requested QoP is not attainable.**

**Explaination:**

The MLP location server reported the following error to the Terminal Location Web Service: The requested QoP is not attainable.

**Response:**

Contact the system administrator.

**SOAM4184E: The MLP location server reported the following error: The requested service is not supported.**

**Explaination:**

The MLP location server reported the following error to the Terminal Location Web Service: The requested service is not supported.

**Response:**

Contact the system administrator.

**SOAM4185E: The MLP location server reported the following error: The protocol element or attribute is not supported.**

**Explaination:**

The MLP location server reported the following error to the Terminal Location Web Service: The protocol element or attribute is not supported.

**Response:**

Contact the system administrator.

**SOAM4186E: The Terminal Location Web Service received the following response: Cancellation of triggered location request.**

**Explaination:**

The Terminal Location Web Service received the following response: Cancellation of triggered location request.

**Response:**

Contact the system administrator.

**SOAM4187E: The MLP location server reported the following error: Unknown MLP protocol.**

**Explaination:**

The MLP location server reported the following error to the Terminal Location Web Service: Unknown MLP protocol.

**Response:**

Contact the system administrator.

**SOAM4188E: The MLP connector could not communicate with the MLP location server.**

**Explaination:**

The MLP connector could not communicate with the MLP location server.

**Response:**

Contact the system administrator.

**SOAM4189E: The report from the MLP location server was corrupted or not valid.**

**Explaination:**

The report from the MLP location server was corrupted or not valid.

**Response:**

No action is required.

**SOAM4300E: A Terminal Location service request failed. A Parlay error occurred.**

**Explaination:**

A Terminal Location service request failed. A Parlay error occurred.

**Response:**

Check the Parlay configuration and the Parlay gateway for errors. Then retry.

**SOAM4302E: A Terminal Location service request failed. The synchronization object timed out.**

**Explaination:**

A Terminal Location service request failed. The synchronization object timed out.

**Response:**

Verify that the Parlay gateway is operational, or increase the timeout value. Then retry.

**SOAM4304E: A Terminal Location service request failed. A Parlay Unauthorized Network error occurred.**

**Explaination:**

A Terminal Location service request failed. A Parlay Unauthorized Network error occurred.

**Response:**

Check the Parlay configuration and retry.

**SOAM4305I: A Terminal Location service request failed A Parlay Position method failure occurred.**

**Explaination:**

A Terminal Location service request failed. A Parlay Position method failure occurred.

**Response:**

Check the Parlay configuration and retry.

**SOAM4308I: A Terminal Location service request failed. The request could not obtain a compatible service manager object from Parlay.**

**Explaination:**

A Terminal Location service request failed. The request could not obtain a compatible service manager object from Parlay.

**Response:**

Check the Terminal Location service name or the Parlay configuration. Then retry.

**SOAM4309I: A Terminal Location service request failed. The operation invoked is not supported.**

**Explaination:**

A Terminal Location service request failed. The operation invoked is not supported.

**Response:**

Make sure the Terminal Location Web service settings match the capabilities of the Parlay gateway. Then retry.

**SOAM4311I: The Terminal Location notification request failed. The specified correlator is already in use.**

**Explaination:**

The Terminal Location notification request failed. The specified correlator is already in use.

**Response:**

Provide a unique correlator for this request, and retry.

**SOAM4312W: Could not create a database entry for a Terminal Location notification request.**

**Explaination:**

Could not create a database entry for a Terminal Location notification request.

**Response:**

Retry the request. If the problem persists, contact the system administrator.

**SOAM4316I: The Terminal Location notification request failed. The specified correlator is not registered.**

**Explaination:**

The Terminal Location notification request failed because. The specified correlator is not registered.

**Response:**

Provide a registered correlator for this request, and retry.

**SOAM4317I: The Terminal Location notification request failed. The specified correlator was not valid.**

**Explaination:**

The Terminal Location notification request failed. The specified correlator was not valid.

**Response:**

Provide a valid correlator for this request, and retry.

**SOAM4318W: A Terminal Location service request failed. The Parlay gateway refused a parameter in the request.**

**Explaination:**

A Terminal Location service request failed. The Parlay gateway refused a parameter in the request.

**Response:**

Check that the parameters of the request are valid, and retry.

**SOAM4319E: A Terminal Location service request failed. The Parlay gateway reported a system failure.**

**Explaination:**

A Terminal Location service request failed. The Parlay gateway reported a system failure.

**Response:**

Check the Parlay gateway for error details.

**SOAM4320E: A Terminal Location service request failed. A Parlay Unauthorized Application error occurred.**

**Explaination:**

A Terminal Location service request failed. A Parlay Unauthorized Application error occurred.

**Response:**

Check the Parlay configuration, and retry.

**SOAM4321E: A Terminal Location service request failed. An unknown Parlay error occurred.**

**Explaination:**

A Terminal Location service request failed. An unknown Parlay error occurred.

**Response:**

Check the Parlay configuration and the Parlay gateway, and retry.

**SOAM4322W: A Terminal Location notification service received a notification from the Parlay gateway that was not valid. The notification was ignored.**

**Explaination:**

A Terminal Location notification service received a notification from the Parlay gateway that was not valid. The notification was ignored.

**Response:**

Check the Parlay configuration and the Parlay gateway.

**SOAM6003E: The delivery status data records could not be retrieved for the following reason: %1.**

**Explaination:**

The delivery status data records could not be retrieved for the following reason: %1.

**Response:**

Contact the system administrator.

**SOAM6006E: The specified correlator provided is in use: %1.**

**Explaination:**

The specified correlator is in use: %1.

**Response:**

Enter a valid value and retry.

**SOAM6015I: Null input value for the href parameter.**

**Explaination:**

No value was specified for the href parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6016I: Empty string input value for the href parameter.**

**Explaination:**

An empty string was specified for the href parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6017I: Null input value for the action parameter.**

**Explaination:**

No value was specified for the action parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6018I: Empty string input value for the action parameter.**

**Explaination:**

An empty string was specified for the action parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6019I: The input string for the action parameter was not valid. Valid values are signal-none, signal-low, signal-medium, signal-high, and signal-delete.**

**Explaination:**

The input string for the action parameter was not valid. Valid values are signal-none, signal-low, signal-medium, signal-high, and signal-delete.

**Response:**

Enter a valid value and retry.

**SOAM6020I: Null input value for the addresses parameter.**

**Explaination:**

No value was specified for the addresses parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6021I: Empty string input value for one of the address parameters.**

**Explaination:**

An empty string was specified for one of the address parameters. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6022I: Null input value for the message parameter.**

**Explaination:**

No value was specified for the message parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6023I: Empty string input value for message parameter.**

**Explaination:**

An empty string was specified for the message parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6025E: The Send SMS data record was not created.**

**Explaination:**

The Send SMS data record was not created.

**Response:**

Contact the system administrator.

**SOAM6026I: Null input value for requestIdentifier parameter.**

**Explaination:**

No value was specified for the requestIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6027I: Empty string input value for requestIdentifier parameter.**

**Explaination:**

An empty string was specified for the requestIdentifier parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6028I: Could not find a delivery status for this request identifier.**

**Explaination:**

Could not find a delivery status for this request identifier.

**Response:**

Enter a valid value and retry.

**SOAM6029E: The delivery status value is unknown.**

**Explaination:**

The delivery status value is unknown.

**Response:**

Contact the system administrator.

**SOAM6030E: The requester ID is null in the MDS context.**

**Explaination:**

The requester ID is null in the MDS context.

**Response:**

Contact the system administrator.

**SOAM6031E: The MDS context is null.**

**Explaination:**

The MDS context is null.

**Response:**

Contact the system administrator.

**SOAM6032E: An error occurred while creating a notification record for the SendWAPpush request.**

**Explaination:**

An error occurred while creating a notification record for the SendWAPpush request.

**Response:**

Contact the system administrator.

**SOAM6055E: Unexpected operation for this method.**

**Explaination:**

This method is not expected to be called by this operation.

**Response:**

Contact the system administrator.

**SOAM6059I: The WAP Push service is disabled.**

**Explaination:**

The WAP Push service is disabled.

**Response:**

Enable the WAPPush service and retry.

**SOAM6060W: Charging is not supported.**

**Explaination:**

Charging is not supported.

**Response:**

Change the service policy setting if you want to support charging.

**SOAM6075E: Could not find the notification data for the correlator.**

**Explaination:**

Could not find the notification data for the correlator.

**Response:**

Contact the system administrator.

**SOAM6076E: There is no connection to the database.**

**Explaination:**

There is no connection to the database.

**Response:**

Contact the system administrator.

**SOAM6077E: An unknown error occurred.**

**Explaination:**

An unknown error occurred. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6079E: A problem occurred while updating the status for the destination.**

**Explaination:**

A problem occurred while updating the status for the destination. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6080E: A problem occurred while finding the segment status for the transaction.**

**Explaination:**

A problem occurred while finding the segment status for the transaction. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6085E: Could not find the notification data for this send transaction ID.**
**Explaination:**

Could not find the notification data for this send transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6086E: Could not perform the notification for this transaction ID.**
**Explaination:**

Could not perform the notification for this transaction ID. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6088E: Could not find the notification data to delete.**
**Explaination:**

Could not find the notification data to delete. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6090E: Could not find expired send data.**
**Explaination:**

Could not find expired send data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6091E: Could not remove expired send data.**
**Explaination:**

Could not remove expired send data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM1593E: Could not remove expired notification data.**
**Explaination:**

Could not remove expired notification data. Check the error log.

**Response:**

Contact the system administrator.

**SOAM6099E: The table is not defined, or the data source could not be created.**
**Explaination:**

The table you are trying to access is not defined, or a problem occurred during initial creation of the data source.

**Response:**

Make sure that the table exists, or check the data source configuration.

**SOAM6100E: The database service is not running, or database authentication failed.** **Explaination:**

The database service is not running, or database authentication failed.

**Response:**

Start the database service or check the authentication credentials. Then retry.

**SOAM6101I: Null input value for the HREF parameter.**
**Explaination:**

No value was specified for the HREF parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6102I: Empty string input value for the HREF parameter.**
**Explaination:**

An empty string was specified for the HREF parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6103I: Null input value for the action parameter.**
**Explaination:**

No value was specified for the action parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6104I: Empty string input value for the action parameter.**
**Explaination:**

An empty string was specified for the action parameter. A value must be specified.

**Response:**

Enter a valid value and retry.

**SOAM6148E: The service policy %1 is not in the MdsContext.**
**Explaination:**

The service policy %1 is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM6149E: The service policy value is not as expected. Service Policy: %1. Policy Value: %2**
**Explaination:**

The service policy value is not as expected. Service Policy: %1. Policy Value: %2

**Response:**

Contact the system administrator.

**SOAM6150: The service policy service.config.target.Aliases is not in the MdsContext.**

**Explaination:**

The service policy service.config.target.Aliases is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM6151E: The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.**

**Explaination:**

The service policy service.config.messaging.ConfirmDelivery is not in the MdsContext.

**Response:**

Contact the system administrator.

**SOAM6152E: No alias names were defined in the Administration Console.**

**Explaination:**

No alias names were defined in the Administration Console.

**Response:**

Contact the system administrator.

**SOAM6153E: An MBeanException occurred while attempting to get an MBean attribute.**

**Explaination:**

An MBeanException occurred while attempting to get an MBean attribute.

**Response:**

Contact the system administrator.

**SOAM6154E: Could not connect to the specified server.**

**Explaination:**

Could not connect to the specified server. A ConnectorException error occurred.

**Response:**

Contact the system administrator.

**SOAM6155E: An internal error occurred in the SMPP Session Manager.**

**Explaination:**

An internal error occurred in the SMPP Session Manager (SmppSessionManager).

**Response:**

Contact the system administrator.

**SOAM6156E: An error occurred while sending the message to the SMSC.**

**Explaination:**

An error occurred while sending the message to the SMSC.

**Response:**

Contact the system administrator.

**SOAM6157E: The Enquire Link Response message was not sent.**

**Explaination:**

The Enquire Link Response message was not sent.

**Response:**

Contact the system administrator.

**SOAM6158E: The Deliver SM Response message was not sent.**

**Explaination:**

The Deliver SM Response message was not sent.

**Response:**

Contact the system administrator.

**SOAM6159E: The Generic Non-Acknowledge message was not sent.**

**Explaination:**

The Generic Non-Acknowledge message was not sent.

**Response:**

Contact the system administrator.

**SOAM6160E: No alias name was defined in the Administration Console. %1**

**Explaination:**

No alias name was defined in the Administration Console. %1

**Response:**

Contact the system administrator.

**SOAM6163E: The specified endpoint URI for notification was not valid.**

**Explaination:**

The specified endpoint URI for notification was not valid.

**Response:**

Enter the correct URI and retry.

**SOAM6164I: The value for the Receipt Request parameter was not null.**

**Explaination:**

A value was specified for the Receipt Request parameter. In order to perform the operation, this parameter must be null.

**Response:**

Deselect the Receipt Request Required option, and retry.

**SOAM6302E: An AttributeNotFoundException occurred while attempting to get an MBean attribute.**

**Explaination:**

An AttributeNotFoundException occurred while attempting to get an MBean attribute.

**Response:**

Contact the system administrator.

**SOAM6303E: An MBeanException occurred while attempting to get an MBean attribute.**

**Explaination:**

An MBeanException occurred while attempting to get an MBean attribute.

**Response:**

Contact the system administrator.

**SOAM6304E: A ReflectionException occurred while attempting to get an MBean attribute.**

**Explaination:**

A ReflectionException occurred while attempting to get an MBean attribute.

**Response:**

Contact the system administrator.

## Messages_SOAS_en_US

This section documents the set of messages that the SOA Service Platform Common components can issue.

**SOAS1000E:{0}:{1} Invalid value for network resource {2} property: {3}**

**Explanation:**

*{0}* is the transaction ID for the traffic shaping common component request. *{1}* is the name of the component that issued the message. *{2}* is the network resource name whose resource specification was being accessed. *{3}* is the network resource property that was missing or invalid. The network resource property was missing or contained an invalid specification property.

**User Response:**

- Check the network resource specification for the specified network resource and verify that the specification contains the property name and a valid value.

**SOAS1500E:{0}{1} Parameter object is null.**

**SOAS1504E:{0}{1} One or more required parameter(s) is invalid.**

**SOAS1505E:{0}{1} Failed to create the {2}. The specified {2} may exist: {3}.**

**SOAS1506E:{0}{1} Failed to update the {2} description. The specified {2} may not exist: {3}.**

**SOAS1507E:{0}{1} Failed to update the {2} status. The specified {2} may not exist: {3}.**

**SOAS1508E:{0}{1} Failed to get the {2} status. The specified {2} may not exist: {3}.**

**SOAS1509E:{0}{1} Failed to remove the {2}. The specified {2} may not exist: {3}.**

**SOAS1510E:{0}{1} Failed to retrieve {2}. The {2} may not exist: {3}.**

**SOAS1511E:{0}{1} Policy name-value pair are invalid, params: {2}.**

**SOAS1512E:{0}{1} Failed to create a SQL Connection.**

**SOAS1513E:{0}{1} SQL statement failed writing to the database.**

**SOAS1514E:{0}{1} SQL Illegal Argument writing to the database.**

**SOAS1515E:{0}{1} SQL Failed to close SQL connection.**

**SOAS1516E:{0}{1} SQL statement error.**

**SOAS1517E:{0}{1} Illegal SQL state exception.**

**SOAS1518E:{0}{1} NotificationRecord not found in database.**

**SOAS1519E:{0}{1} SQL error reading a row or colum from the DB query.**

**SOAS1520E:{0}{1} Invalid input value {2} for message part {3}.**

**SOAS1521I:{0}{1} An invalid message part was received: [service = {2},[requester = {3}],[correlator = {4}]**

**Invalid input value %1 for message part %2.**

**SOAS1523E:{0}{1} A service error occurred. Failure reason is {2}.**

**SOAS1524I:{0}{1} Unable to remove notification records: [service = {2}],[requester = {3}],[correlator = {4}],[reason = {5}],[details = {6}]**

**A service error occurred. Failure reason is %1.**

**SOAS1526E:{0}{1} Notification records not found.**

**SOAS1527I:{0}{1} Unable to find notification records: [service = {2}],[requester = {3}],[correlator = {4}]**

**Notification records not found.**

**SOAS1529E:{0}{1} Duplicate Record found.**

**SOAS1530E:{0}{1} Unable to close database connections.**

**SOAS1531E:{0}{1} GlobalID Notification records not found.**

**SOAS1532E:{0}{1} Start and End Notification records not found.**

**SOAS1533E:{0}{1} Target and Criteria Notification records not found.**

**SOAS1534E:{0}{1} Terminate All Notification records not found.**

**SOAS1535E:{0}{1} PortType and EndPoint Notification records not found.**

**SOAS1536E:{0}{1} FindAll Notifications returned no records.**

**SOAS1537E:{0}{1} ResetAll Notifications records not found.**

**SOAS1538E:{0}{1} NotificationAdministrationSupport Interface not supported.**

**SOAS1539E:{0}{1} Notification Manager Server error {2}.**

**SOAS2000E:{0}:{1} Invalid Delivery Method Specified [{2}].**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the delivery method specified. An invalid delivery method was specified in the message.

**User Response:**

- None: possibly a programming error.

**SOAS2001E:{0}:{1} Invalid message category received [{2}].**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the message category that was received An message with an invalid message category was received.

**User Response:**

- None: possibly a programming error.

**SOAS2002E:{0}:{1} Invalid message operation received [{2}], for category {3}.**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the name of the operation. *{3}* is the operation category. An invalid message operation was received.

**User Response:**

- None: possibly a programming error.

**SOAS2003E:{0}:{1} Invalid JMS Message type received [{2}], expected javax.jms.ObjectMessage .**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the class name of the invalid JMS message type. An invalid JMS message type was received when an javax.jms.ObjectMessage type was expected.

**User Response:**

- None: possibly a programming error.

**SOAS2004E:{0}:{1} Error extracting data from JMS message.**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. An exception occurred while trying to extracting data from a JMS message.

**User Response:**

- None: possibly a programming error.

**SOAS2005E:{0}:{1} Invalid Callback endpoint value specified [{2}].**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the callback endpoint value. An invalid callback endpoint value was specified.

**User Response:**

- Verify that the callback endpoint was correctly configured.

**SOAS2006E:{0}:{1} Empty or Null ESB endpoint URI specified.**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. An empty or null ESB endpoint URI was specified.

**User Response:**

- None.

**SOAS2007E:{0}:{1} Invalid numeric timeout value specified [{2}].**

**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the numeric timeout value that was specified. An invalid numeric timeout value was specified for the endpoint component.

**User Response:**

- Verify that the timeout value specified for the endpoint component has been correctly configured.

**SOAS2008E:{0}:{1} Throwable exception caught, unable to issue callback, error: [{2}].**
**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the message from the throwable exception. Throwable exception caught, unable to issue a callback because the SPNotificationExtension was not available.

**User Response:**

- None: review exception message for possible causes.

**SOAS2009E:{0}:{1} Invalid ESB endpoint URI specified [{2}].**
**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the the endpoint URI that was specified. An invalid ESB endpoint URI was specified.

**User Response:**

- Verify that the endpoint has been correctly configured.

**SOAS2010E:{0}:{1} Empty or Null Delivery Method specified.**
**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. An empty or null delivery method was specified.

**User Response:**

- None: possibly a programming error.

**SOAS2011E:{0}:{1} Error attempting to delivery notification, error: [{2}].**
**Explanation:**

*{0}* is the transaction ID for the pxnotify common component request. *{1}* is the name of the component that issued the message. *{2}* is the message from the exception that occurred. An exception occurred while attempting to delivery notification to an operation category.

**User Response:**

- None: review exception message for possible causes.

**SOAS3000E:{0}:{1} Request object is null.**
**Explanation:**

*{0}* is the transaction ID for the usage records common component request. *{1}* is the name of the usage records common component. The request object received is null.

**User Response:**

- None: possibly a programming error.

**SOAS3001E:{0}:{1} Write usage record failed. Check trace for details.**
**Explanation:**

*{0}* is the transaction ID for the usage records common component request. *{1}* is the name of the usage records common component. Unable to write a usage record.

**User Response:**

- Check the trace log for possible causes.

**SOAS3002E:{0}:{1} Read usage record failed. Check trace for details.**

**Explanation:**

*{0}* is the transaction ID for the usage records common component request. *{1}* is the name of the usage records common component. Unable to read a usage record.

**User Response:**

- Check the trace log for possible causes.

**SOAS3003E:{0}:{1} Could not access datasource.**

**Explanation:**

*{0}* is the transaction ID for the usage records common component request. *{1}* is the name of the usage records common component. Unable to access the datasource.

**User Response:**

- Check the trace log for possible causes.

**SOAS3004E:{0}:{1} Could not get a database connection.**

**Explanation:**

*{0}* is the transaction ID for the usage records common component request. *{1}* is the name of the usage records common component. Unable to get a database connection.

**User Response:**

- Check the trace log for possible causes.

**SOAS3005E:{0}:{1} Unable to close database connection.**

**Explanation:**

*{0}* is the transaction ID for the usage records common component request. *{1}* is the name of the usage records common component. Unable to close a database connection.

**User Response:**

- Check the trace log for possible causes.

**SOAS3006E:{0}:{1} Invalid usage record field: {2}.**

**SOAS3007E:{0}:{1} Error writing usage record to database: {2}**

**SOAS4001E:{0}{1} Invalid input parameter, params: {2}**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter value in the input Invalid input parameter.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4002E:{0}{1} Operation failed, the parameter {2} is not valid, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the parameter is not valid

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4003E:{0}{1} Operation failed, the specified {2} already exists, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input The specified parameter already exists.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4004E:{0}{1} Operation failed, the specified {2} does not exist, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter does not exist

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4005E:{0}{1} Operation failed, the type of specified {2} is not valid for creating {3}, params: {4}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter type in the input *{4}* is the invalid parameter value in the input Operation failed, the specified parameter is not valid in the creation operation

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4006E:{0}{1} Operation failed, the specified {2} is the same with the existing one, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter is the same with the existing one

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4007E:{0}{1} Operation failed, the predefined {2} cannot be created or removed, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter is predefined value which can not be created or removed

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4008E:{0}{1} Operation failed, the specified {2} cannot be removed because it contains children, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter cannnot be removed becaue it contains children

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4009E:{0}{1} Operation failed, the specified {2} may not contain {3}, params: {4}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter type in the input *{4}* is the invalid parameter value in the input Operation failed, the specified parameter may not contain the specified type.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4010E:{0}{1} Operation failed, the specified {2} is not an individual {2}, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter is not an individual type

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4011E:{0}{1} Operation failed, the specified {2} or one of its parents is disabled, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter or one of its parent is disabled

**User Response:**

- Make sure the item and its parent are enabled, and then try again

**SOAS4012E:{0}{1} Operation failed, the specified Subscription cannot be created because a subscription to a different implementation of the same service already exists, params: {2}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter value in the input Operation failed, the specified Subscription cannot be created because a subscription to a different implementation of the same service already exists.

**User Response:**

- (1) Remove existing subscription, or (2)Verify the input parameters, and use valid parameter(s), then try again

**SOAS4013E:{0}{1} Operation failed, the type of specified {2} is not a group, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter is not a group.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4014E:{0}{1} Operation failed, the specified {2} is not editable, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter is not editable.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4015E:{0}{1} Operation failed, the specified {2} and {3} cannot be same, params: {4}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter in the input *{4}* is the invalid parameter value in the input Operation failed, the specified parameter may not contain the specified type.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4016E:{0}{1} Operation failed, the specified policy value does not match the policy type, params: {2}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter value in the input Operation failed, the specified policy value does not match the policy type.

**User Response:**

- Make sure policy value and type are compatible,then try again

**SOAS4017E:{0}{1} Operation failed, the type of specified {2} is neither an individual service nor a service group, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the type of specified parameter is neither an individual service nor a service group.

**User Response:**

- Make sure the parameter is a individual service or service group, then try again

**SOAS4018E:{0}{1} Operation failed, the specified {2} can not be the child item or one of its descendants, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter can not be the child item or one of its descendants.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4019E:{0}{1} Operation failed, the parent of predefined {2} ALL cannot be updated, params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the parent of predefined ALL cannot be updated.

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4020E:{0}{1} Operation failed, the value of the parameter {2} exceeds the maximum length allowed (250 characters), params: {3}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the value of the parameter exceeds the maximum length allowed (250 characters).

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4021E:{0}{1} Operation failed, the specified {2} contains an invalid occurrence of the escape character (the escape character can only appear in the pattern string if it is followed by itself, %, or _), params: {2}.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". *{2}* is the invalid parameter(s) in the input *{3}* is the invalid parameter value in the input Operation failed, the specified parameter contains an invalid occurrence of the escape character (the escape character can only appear in the pattern string if it is followed by itself, %, or _).

**User Response:**

- Verify the input parameters, and use valid parameter(s) to try again

**SOAS4496E:{0}{1} Failed to get data source.**

**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is "ServicePolicyManager". Failed to get data source.

**User Response:**

- Datasoure is not configure properly, please contact administrator

**SOAS4497E:{0}{1} Failed to get database connection.**
**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. Failed to get database connection.

**User Response:**

- Database is not running or connection parameter is not specified properly, please contact administrator

**SOAS4498E:{0}{1} Internal database error.**
**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. Internal database error.

**User Response:**

- Database is not in consistent state or a database error occurs, please contact administrator

**SOAS4499E:{0}{1} Internal service error.**
**Explanation:**

*{0}* Empty, reserved for transaction ID. *{1}* is the name of the component that issued the message, which is ″ServicePolicyManager″. Internal service error.

**User Response:**

- Please contact administrator

**SOAS4500E: An internal error occurred; the Service Policy Manager Console was unable to fulfill the request.**
**Explanation:**

An internal problem prevented the application from successfully processing your request.

**User Response:**

- Please contact the administrator and/or examine the system logs.
- Diagnosing the problem will require enabling diagnostic trace.

**SOAS4501E: Unable to contact the Service Policy Manager Console.**
**Explanation:**

The web browser either could not contact the application, or it did not receive a timely response from the application.

**User Response:**

- Please check connectivity with the application server.
-
- After checking connectivity, please attempt to re-login to the application.
- Diagnosing the problem may require enabling the diagnostic trace.

**SOAS4502E: Unauthorized to access the Service Policy Manager Console.**
**Explanation:**

The server reported that you are not authorized to access this content or function.

**User Response:**

- Please verify the your username and password.
- Please verify that you are authorized to access the application.
- Please contact the administrator and/or examine the system logs.

**SOAS4503E: A malformed request was reported; the Service Policy Manager Console was unable to fulfill the request.**

**Explanation:**

The application reports that you've issued a malformed request that it could not process.

**User Response:**

- Please contact the administrator and/or examine the system logs.
- Diagnosing the problem will require enabling diagnostic trace.

## Messages_SOAX_en_US

This section documents the set of messages that the SOA Service Implementation components can issue.

**SOAX0001E:{0}{1} {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the component that issued the message. A Parlay X Web Service fault condition occurred.

**User Response:**

- Review the Parlay X Web Service fault documentation for more information.

**SOAX0002I:{0}{1} Operation {2} must be invoked within an existing session: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the Web Service operation invoked by the client. *{3}* contains details about the specific session object and value (example: ″HttpSession == null″) behind receipt of this message. An incoming Web Service request needed to within a preexisting session, but was not. Because of the lack of a preexisting session, the request was rejected.

**User Response:**

- Ensure the Web services client is maintaining sessions across multiple Web Service requests.
- Ensure the Web services client is following the expected use case/method invocation order. Example: invoking StartNotification before EndNotification.

**SOAX0004I:{0}{1} Policy {2} with value {3} is malformed; it must follow the syntax and semantics of the configuration setting {4}**

**Explanation:**

*{0}* is the transaction ID for the Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the policy provided by the client. *{3}* is the value of the policy provided by the

client. *{4}* is the name of the configuration setting corresponding to the policy. An incoming Web Service request specified a policy value that was invalid according to the syntax and semantics of the corresponding configuration setting. Because it could not service the request's malformed policy value, the service implementation rejected the request.

**User Response:**

- Ensure the Web Services client is providing the intended policy
- Ensure the Web Services client is providing a value that is valid for the setting according to the administrative console. Refer to the administrative console's help for details on each of the settings.

**SOAX0005I:{0}{1} Policy {2} with value {3} is invalid; its value must be more restrictive than the configuration setting {4} which has value {5}**

**Explanation:**

*{0}* is the transaction ID for the Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the policy provided by the client. *{3}* is the value of the policy provided by the client. *{4}* is the name of the configuration setting corresponding to the policy. *{5}* is the current value the configuration setting. An incoming Web Service request specified a policy value that was less restrictive than the corresponding configuration setting's value. Incoming policy values must be more restrictive than the settings they override. This is to ensure that policy does not violate installation-specific settings. Because it could not service the request's invalid policy value, the service implementation rejected the request.

**User Response:**

- Ensure the Web Services client is providing the intended policy
- Ensure the Web Services client is providing a value that is valid for the setting according to the administrative console. Refer to the administrative console's help for details on each of the settings.
- Ensure the Web Services client is providing a value that is more restrictive than the current administrative console setting. The notion of ″more restrictive″ varies on a per-setting basis.

**SOAX0006I:{0}{1} Policy {2} with value {3} is invalid; it attempts to overwrite the read-only configuration setting {4}.**

**Explanation:**

*{0}* is the transaction ID for the Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the policy provided by the client. *{3}* is the value of the policy provided by the client. *{4}* is the name of the configuration setting corresponding to the policy. An incoming Web Service request specified a policy value that attempts to override a read-only configuration setting. Because it could not service the request's invalid policy value, the service implementation rejected the request.

**User Response:**

- Ensure the Web Services client is providing the intended policy.
- Some configuration settings cannot be overridden. See the administrative console help for more details on the given configuration setting.

**SOAX0100E:{0}{1} Admission control rejected request; it would exceed per service limit**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The Admission Control component rejected a request because accepting it would exceed the currently defined per service limits.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Admission Control's per service limits

**SOAX0101W:{0}{1} Admission control rejected request; it would exceed per service limit**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The Admission Control component rejected a request because accepting it would exceed the currently defined per service limits.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Admission Control's per service limits

**SOAX0102I:{0}{1} Admission control rejected request; it would exceed per service limit**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The Admission Control component rejected a request because accepting it would exceed the currently defined per service limits.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Admission Control's per service limits

**SOAX0103E:{0}{1} Admission control rejected request; it would exceed per operation limit**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The Admission Control component rejected a request because accepting it would exceed the currently defined per operation limits.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Admission Control's per service, per operation limits

**SOAX0104W:{0}{1} Admission control rejected request; it would exceed per operation limit**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The Admission Control component rejected a request because accepting it would exceed the currently defined per operation limits.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Admission Control's per service, per operation limits

**SOAX0105I:{0}{1} Admission control rejected request; it would exceed per operation limit**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The Admission Control component rejected a request because accepting it would exceed the currently defined per operation limits.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Admission Control's per service, per operation limits

**SOAX0106E:{0}{1} Traffic shaping rejected request; it would exceed limits for resource {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource whose limits would be exceeded. The Traffic Shaping component rejected a request because accepting it would exceed limits for the specified resource.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Traffic Shaping's per resource, per operation limits

**SOAX0107W:{0}{1} Traffic shaping rejected request; it would exceed limits for resource {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource whose limits would be exceeded. The Traffic Shaping component rejected a request because accepting it would exceed limits for the specified resource.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Traffic Shaping's per resource, per operation limits

**SOAX0108I:{0}{1} Traffic shaping rejected request; it would exceed limits for resource {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource whose limits would be exceeded. The Traffic Shaping component rejected a request because accepting it would exceed limits for the specified resource.

**User Response:**

- If you know the per service limits are correct, no action is required
- Otherwise, correct Traffic Shaping's per resource, per operation limits

**SOAX0109E:{0}{1} Unable to register notification; Notification management reported fault: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the notification management component. The Notification Management component reported an error while trying to register a new notification.

**User Response:**

- Examine the reported fault and detail message. Consult the Notification Management component documentation for more information.

**SOAX0110W:{0}{1} Unable to register notification; Notification management reported fault: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the notification management component. The Notification Management component reported an error while trying to register a new notification.

**User Response:**

- Examine the reported fault and detail message. Consult the Notification Management component documentation for more information.

**SOAX0111I:{0}{1} Unable to register notification; Notification management reported fault: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the notification management component. The Notification Management component reported an error while trying to register a new notification.

**User Response:**

- Examine the reported fault and detail message. Consult the Notification Management component documentation for more information.

**SOAX0112E:{0}{1} Unable to remove notification; Notification management reported fault: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the notification management component. The Notification Management component reported an error while trying to remove an existing notification.

**User Response:**

- Examine the reported fault and detail message. Consult the Notification Management component documentation for more information.

**SOAX0113W:{0}{1} Unable to remove notification; Notification management reported fault: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the notification management component. The Notification Management component reported an error while trying to remove an existing notification.

**User Response:**

- Examine the reported fault and detail message. Consult the Notification Management component documentation for more information.

**SOAX0114I:{0}{1} Unable to remove notification; Notification management reported fault: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the notification management component. The Notification Management component reported an error while trying to remove an existing notification.

**User Response:**

- Examine the reported fault and detail message. Consult the Notification Management component documentation for more information.

**SOAX0115E:{0}{1} Usage record component reported fault while writing record: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the usage record component. The Usage Record component reported an error while trying to write a record.

**User Response:**

- Examine the reported fault and detail message. Consult the Usage Record component documentation for more information.

**SOAX0116W:{0}{1} Usage record component reported fault while writing record: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the usage record component. The Usage Record component reported an error while trying to write a record.

**User Response:**

- Examine the reported fault and detail message. Consult the Usage Record component documentation for more information.

**SOAX0117I:{0}{1} Usage record component reported fault while writing record: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the usage record component. The Usage Record component reported an error while trying to write a record.

**User Response:**

- Examine the reported fault and detail message. Consult the Usage Record component documentation for more information.

**SOAX0118E:{0}{1} Usage record component reported fault while reading recordID {2}: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the existing record that the service implementation is trying to read. *{3}* is the fault reported by the usage record component. The Usage Record component reported an error while trying to read an existing record.

**User Response:**

- Examine the reported fault and detail message. Consult the Usage Record component documentation for more information.

**SOAX0119W:{0}{1} Usage record component reported fault while reading recordID {2}: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the existing record that the service implementation is trying to read. *{3}* is the fault reported by the usage record component. The Usage Record component reported an error while trying to read an existing record.

**User Response:**

- Examine the reported fault and detail message. Consult the Usage Record component documentation for more information.

**SOAX0120I:{0}{1} Usage record component reported fault while reading recordID {2}: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the existing record that the service implementation is trying to read. *{3}* is the fault reported by the usage record component. The Usage Record component reported an error while trying to read an existing record.

**User Response:**

- Examine the reported fault and detail message. Consult the Usage Record component documentation for more information.

**SOAX0121E:{0}{1} Privacy component reported fault while retrieving information: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the privacy component. The Privacy component reported an error while trying to process a privacy information request.

**User Response:**

- Examine the reported fault and detail message. Consult the Privacy component documentation for more information.

**SOAX0122W:{0}{1} Privacy component reported fault while retrieving information: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault reported by the privacy component. The Privacy component reported an error while trying to process a privacy information request.

**User Response:**

- Examine the reported fault and detail message. Consult the Privacy component documentation for more information.

**SOAX0123I:{0}{1} Privacy component reported fault while retrieving information: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault

reported by the privacy component. The Privacy component reported an error while trying to process a privacy information request.

**User Response:**

- Examine the reported fault and detail message. Consult the Privacy component documentation for more information.

**SOAX0124E:{0}{1} No admission control limits configured for service {2} operation {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the service that no admission control limits are defined for. *{3}* is the name of the operation that no admission control limits are defined for. The admission control component is rejecting incoming requests because it has not been configured for this given service and operation.

**User Response:**

- Configure the admission control component within appropriate limits for this service and operation.

**SOAX0125W:{0}{1} No admission control limits configured for service {2} operation {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the service that no admission control limits are defined for. *{3}* is the name of the operation that no admission control limits are defined for. The admission control component is rejecting incoming requests because it has not been configured for this given service and operation.

**User Response:**

- Configure the admission control component within appropriate limits for this service and operation.

**SOAX0126I:{0}{1} No admission control limits configured for service {2} operation {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the service that no admission control limits are defined for. *{3}* is the name of the operation that no admission control limits are defined for. The admission control component is rejecting incoming requests because it has not been configured for this given service and operation.

**User Response:**

- Configure the admission control component within appropriate limits for this service and operation.

**SOAX0127E:{0}{1} Traffic shaping not configured for resource {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource without a traffic shaping configuration. The traffic shaping component is rejecting incoming requests because the given resource has not been configured.

**User Response:**

- Configure the traffic shaping component within appropriate limits for this resource.

**SOAX0128W:{0}{1} Traffic shaping not configured for resource {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource without a traffic shaping configuration. The traffic shaping component is rejecting incoming requests because the given resource has not been configured.

**User Response:**

- Configure the traffic shaping component within appropriate limits for this resource.

**SOAX0129I:{0}{1} Traffic shaping not configured for resource {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource without a traffic shaping configuration. The traffic shaping component is rejecting incoming requests because the given resource has not been configured.

**User Response:**

- Configure the traffic shaping component within appropriate limits for this resource.

**SOAX0130E:{0}{1} Traffic shaping reports an invalid resource specification: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource with an invalid traffic shaping configuration. The traffic shaping component is rejecting incoming requests because the given resource has been configured incorrectly.

**User Response:**

- Repair the traffic shaping configuration for this resource.

**SOAX0131W:{0}{1} Traffic shaping reports an invalid resource specification: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource with an invalid traffic shaping configuration. The traffic shaping component is rejecting incoming requests because the given resource has been configured incorrectly.

**User Response:**

- Repair the traffic shaping configuration for this resource.

**SOAX0132I:{0}{1} Traffic shaping reports an invalid resource specification: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the resource with an invalid traffic shaping configuration. The traffic shaping component is rejecting incoming requests because the given resource has been configured incorrectly.

**User Response:**

- Repair the traffic shaping configuration for this resource.

**SOAX0133E:{0}{1} Parlay X notification delivery unable to deliver notification: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the error detail from the Parlay X notification delivery component. The Parlay X notification delivery component could not deliver the requested notification.

**User Response:**

- Examine the error detail and consult the Parlay X notification delivery component documentation to resolve.

**SOAX0134W:{0}{1} Parlay X notification delivery unable to deliver notification: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the error detail from the Parlay X notification delivery component. The Parlay X notification delivery component could not deliver the requested notification.

**User Response:**

- Examine the error detail and consult the Parlay X notification delivery component documentation to resolve.

**SOAX0135I:{0}{1} Parlay X notification delivery unable to deliver notification: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the error detail from the Parlay X notification delivery component. The Parlay X notification delivery component could not deliver the requested notification.

**User Response:**

- Examine the error detail and consult the Parlay X notification delivery component documentation to resolve.

**SOAX0150E:{0}{1} Error invoking the admission control component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the admission control common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the admission control component endpoint
- Ensure that the admission control component is accepting requests

**SOAX0151W:{0}{1} Error invoking the admission control component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the admission control common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the admission control component endpoint
- Ensure that the admission control component is accepting requests

**SOAX0152I:{0}{1} Error invoking the admission control component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the admission control common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the admission control component endpoint
- Ensure that the admission control component is accepting requests

**SOAX0153E:{0}{1} Error invoking the traffic shaping component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the traffic shaping common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the traffic shaping component endpoint
- Ensure that the traffic shaping component is accepting requests

**SOAX0154W:{0}{1} Error invoking the traffic shaping component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the traffic shaping common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the traffic shaping component endpoint
- Ensure that the traffic shaping component is accepting requests

**SOAX0155I:{0}{1} Error invoking the traffic shaping component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the traffic shaping common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the traffic shaping component endpoint
- Ensure that the traffic shaping component is accepting requests

**SOAX0156E:{0}{1} Error invoking the notification management component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the notification management common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the notification management component endpoint.
- Ensure that the notification management component is accepting requests.

**SOAX0157W:{0}{1} Error invoking the notification management component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the notification management common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the notification management component endpoint.
- Ensure that the notification management component is accepting requests.

**SOAX0158I:{0}{1} Error invoking the notification management component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the notification management common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the notification management component endpoint.
- Ensure that the notification management component is accepting requests.

**SOAX0159E:{0}{1} Error invoking the Parlay X notification delivery component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the Parlay X Notification Delivery common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the notification delivery component endpoint.
- Ensure that the notification delivery component is accepting requests.

**SOAX0160W:{0}{1} Error invoking the Parlay X notification delivery component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the Parlay X Notification Delivery common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the notification delivery component endpoint.
- Ensure that the notification delivery component is accepting requests.

**SOAX0161I:{0}{1} Error invoking the Parlay X notification delivery component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the Parlay X Notification Delivery common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the notification delivery component endpoint.
- Ensure that the notification delivery component is accepting requests.

**SOAX0162E:{0}{1} Error invoking the usage record component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while

invoking the usage record common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the usage record component endpoint.
- Ensure that the usage record component is accepting requests.

**SOAX0163W:{0}{1} Error invoking the usage record component: {2}**
**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the usage record common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the usage record component endpoint.
- Ensure that the usage record component is accepting requests.

**SOAX0164I:{0}{1} Error invoking the usage record component: {2}**
**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the usage record common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the usage record component endpoint.
- Ensure that the usage record component is accepting requests.

**SOAX0165E:{0}{1} Error invoking the privacy component: {2}**
**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the privacy common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the privacy component endpoint.
- Ensure that the privacy component is accepting requests.

**SOAX0166W:{0}{1} Error invoking the privacy component: {2}**
**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while

invoking the privacy common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the privacy component endpoint.
- Ensure that the privacy component is accepting requests.

**SOAX0167I:{0}{1} Error invoking the privacy component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the privacy common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the privacy component endpoint.
- Ensure that the privacy component is accepting requests.

**SOAX0168E:{0}{1} Error invoking the faults and alarms component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the faults and alarms common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the faults and alarms component endpoint.
- Ensure that the faults and alarms component is accepting requests.

**SOAX0169W:{0}{1} Error invoking the faults and alarms component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while invoking the faults and alarms common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the faults and alarms component endpoint.
- Ensure that the faults and alarms component is accepting requests.

**SOAX0170I:{0}{1} Error invoking the faults and alarms component: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the exception that was returned while invoking the common component. The service implementation encountered an unexpected exception while

invoking the faults and alarms common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the faults and alarms component endpoint.
- Ensure that the faults and alarms component is accepting requests.

**SOAX0200I:{0}{1} Check that endpoint is properly configured. Ensure configured endpoint is accepting requests.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation encountered an unexpected exception while invoking a common component. This may be due to invalid endpoint information, network connectivity problems, remote server errors, etc.

**User Response:**

- Verify the configuration for the relevant component endpoint.
- Ensure that the component is accepting requests.

**SOAX0201I:{0}{1} Configure admission control limits for this service and operation.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The admission control component reported that this service or operation has no currently configured admission limits.

**User Response:**

- Create the admission configuration for this service or operation, or both.

**SOAX0202I:{0}{1} Check that traffic shaping component has been properly configured for this resource.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The traffic shaping component reported that this resource has no currently configured traffic limits.

**User Response:**

- Create the traffic shaping configuration for this resource.

**SOAX0203I:{0}{1} Correct this resource's specification in the traffic shaping component.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The traffic shaping component reported that this resource has an error in its configuration.

**User Response:**

- Verify the traffic shaping configuration for this resource.

**SOAX0204I:{0}{1} If performance is impacted, reconfigure admission control limits for this service and operation.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The admission control component rejected a request because accepting the request would have violated the configured service and operation limits.

**User Response:**

- Verify the admission control configuration for this resource.
- If the configuration is correct, this message indicates normal operation for a rejected request.
- If the configuration is not correct, adjust the service and operation admission limits as appropriate. See the admission control documentation for more information.

**SOAX0205I:{0}{1} Additional detail provided for fault/alarm {2}: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the fault or alarm code for which more information is being provided. *{3}* is the additional detail related to the fault or alarm. The service implementation is providing detail relevant to a recently generated fault or alarm.

**User Response:**

- More information about the condition can be found by looking for the fault or alarm code provided.

**SOAX0221I:{0}{1} Policy [{2}] not found.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of the policy. The service implementation could not find the named policy in the SOAP headers of the Web Service request. The ESB component adds policies to the SOAP headers before they are passed to the Service Platform component.

**User Response:**

- Actions are dependent on the operator's implementation of the policy.

**SOAX0222W:{0}{1} Invalid value [{2}] for policy [{3}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the value of the policy. *{3}* is the name of the policy. The value for the named policy is not valid.

**User Response:**

- Correct the policy's value in the appropriate policy database table.
- If the policy's value is correct in the policy database tables, then this is an internal processing error.

**SOAX0223W:{0}{1} Attribute [{2}] not found.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the value of the attribute in the CONFIGPROPERTIES database table. The service implementation could not find the named attribute in the CONFIGPROPERTIES database table.

**User Response:**

- Verify that you created the CONFIGPROPERTIES table. If not, run the appropriate DDL files.
- If the CONFIGPROPERTIES table exists, try to open the configuration pages for the service implementation in the Administrative console and configure the attribute.
- If the service implementation cannot be configured through the Administrative console, the table may be corrupted and you may need to recreate it using the appropriate DDL files.

**SOAX0224W:{0}{1} Invalid value [{2}] for attribute [{3}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the value of the attribute. *{3}* is the name of the attribute. The value for the named attribute is not valid.

**User Response:**

- Open the configuration pages for the service implementation in the Administrative console and configure the attribute.

**SOAX0225I:{0}{1} Programming error. [{2}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is an internal processing message containing internal processing values. The service implementation detected an unexpected internal processing error.

**User Response:**

- None: an internal processing error occurred.

**SOAX0226I:{0}{1} Requester name null or zero-length.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation received a request with a null or empty requester. Functionality relying upon a valid requester name, such as privacy, may be incorrect. This may be caused by a Web Service client that is not authenticating itself with the service platform.

**User Response:**

- Ensure that the Web Service client is authenticating itself with the service platform

**SOAX0227I:{0}{1} Unable to retrieve current server name.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation could not determine the current Application server name. Faults and alarms may have a NullServerName in their source field.

**User Response:**

- None: the condition is due to an internal error.

**SOAX0228I:{0}{1} Unable to retrieve current host name.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation could not determine the current host name. Faults and alarms may have a NullHostName in their source field.

**User Response:**

- Ensure that the service platform has hostnames configured

**SOAX0229I:{0}{1} Unable to retrieve current EAR name.**
**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation could not determine the current Enterprise Application Resource (EAR) name. Faults and alarms may have a NullEarName in their source field.

**User Response:**

- None: the condition is due to an internal error.

**SOAX0250E:{0}{1} Invalid structure or content in {2} request: {3}**

**SOAX0251W:{0}{1} Invalid structure or content in {2} request: {3}**

**SOAX0252I:{0}{1} Invalid structure or content in {2} request: {3}**

**SOAX0253E:{0}{1} Duplicate notification management entry: {2}**

**SOAX0254W:{0}{1} Duplicate notification management entry: {2}**

**SOAX0255I:{0}{1} Duplicate notification management entry: {2}**

**SOAX0256E:{0}{1} Service error occurred while processing the notification management request: {2}**

**SOAX0257W:{0}{1} Service error occurred while processing the notification management request: {2}**

**SOAX0258I:{0}{1} Service error occurred while processing the notification management request: {2}**

**SOAX0259E:{0}{1} Notification management entry not found: {2}**

**SOAX0260W:{0}{1} Notification management entry not found: {2}**

**SOAX0261I:{0}{1} Notification management entry not found: {2}**

**SOAX0270E:{0}{1} Invalid input value {2} for message part {3}.**

**SOAX0271I:{0}{1} An invalid message part was received: {2}**

**Invalid input value %1 for message part %2.**

**SOAX0273E:{0}{1} A service error occurred. Failure reason is {2}.**

**SOAX0274I:{0}{1} Unable to remove notification records: {2}**

**A service error occurred. Failure reason is %1.**

**SOAX0276E:{0}{1} Notification records not found.**

**SOAX0277I:{0}{1} Unable to find notification records: {2}**

**Notification records not found.**

**SOAX0500E:{0}{1} An unexpected exception occurred.**
**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. An unexpected exception occurred in the service implementation that will halt processing of the Web Service request.

**User Response:**

- Verify that the service implementation modules started without errors.
- Verify that you have correctly configured the service implementation application.
- If the service implementation has started without errors and has been correctly configured, it may be due to an internal error processing the data in the Web Service request.

**SOAX0501W:{0}{1} An unexpected exception occurred. Processing will continue.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. An unexpected exception occurred in the service implementation, but processing of the Web Service request will continue if possible.

**User Response:**

- Verify that the service implmentation modules started without errors.
- Verify that you have correctly configured the service implementation application.
- If the service implementation has started without errors and has been correctly configured, it may be due to an internal error processing the data in the Web Service request.

**SOAX0502E:{0}{1} Unable to create a SIP application session.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation could not create a SIP application session while processing a makeCall operation. It is likely that one of the service implementation SIP servlets did not start successfully.

**User Response:**

- Verify that the service implementation modules started without errors.
- Verify that you have correctly configured the service implementation application.

**SOAX0503E:{0}{1} Unable to obtain a call identifier from the SIP application session.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. The service implementation created a SIP application session while processing a makeCall operation, but could not get a call identifier from the session. It is likely one of the service implementation SIP servlets did not start successfully.

**User Response:**

- Verify that the service implementation modules started without errors.
- Verify that you have correctly configured the service implementation application.

**SOAX0551I:{0}{1} Check SIP servlet is started, check S-CSCF routing header is configured correctly.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. This message contains the suggested action to take when there is an error sending a SIP message. It is associated with messages SOAX0552, SOAX0553, and SOAX0554.

**User Response:**

- Take the suggested action in this message.

**SOAX0552E:{0}{1} Error sending SIP messages: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. One of the service implementation's SIP servlets received a 6XX error.

**User Response:**

- Take the suggested action in message SOAX0551.

**SOAX0553W:{0}{1} Error sending SIP messages: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. One of the service implementation's SIP servlets received a 6XX error.

**User Response:**

- Take the suggested action in message SOAX0551.

**SOAX0554I:{0}{1} Error sending SIP messages: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. One of the service implementation's SIP servlets received a 6XX error.

**User Response:**

- Take the suggested action in message SOAX0551.

**SOAX0555W:{0}{1} Policy or attribute configuration error: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* contains a policy or attribute and its current value. A policy or its corresponding attribute contains an invalid value for the expected type. For example, the value contains an alphanumeric value instead of an expected numeric value.

**User Response:**

- Take the suggested action in message SOAX0556.

**SOAX0556I:{0}{1} Verify that the policy or attribute is correctly configured.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. This message contains the suggested action to take when there is an error validating a policy or attribute value. It is associated with message SOAX0555.

**User Response:**

- Take the suggested action in message SOAX0556.

**SOAX0557W:{0}{1} Unable to add P-Asserted-Identity header to outbound request. Requester ID [ {2} ] could not be used to create a valid P-Asserted-Identity value.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the requester ID. A valid p-asserted-identity value could not be created from the requester ID that the ESB component passed in a SOAP header.

**User Response:**

- None. Programming error.

**SOAX1001I:{0}{1} Data source access error. Verify that the data source is running or tables exist.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message.

**User Response:**

- Ensure the data source is up and running.

**SOAX1003I:{0}{1} Duplicate Key Exception from Database, check input value.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message.

**User Response:**

- Verify that the correlator is duplicate.

**SOAX1005E:{0}{1} VerifyPrivacyPermits results false:{2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message.

**User Response:**

- Verify if the privacy for this request is correct.

**SOAX1015I:{0}{1} Notification delivery status: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* Notification delivery status.

**User Response:**

- None.

**SOAX1017E:{0}{1} Exception while {2}.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the operation name while the exception occurs.

**User Response:**

- Review the log file and correct the cause for the exception.

**SOAX1018W:{0}{1} Use default values for {2}.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the default value.

**User Response:**

- None.

**SOAX1020E:{0}{1} URI has IllegalArgument: {2}.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* URI argument.

**User Response:**

- Check the URI for correct syntax.

**SOAX1033W:{0}{1} Can not retrieve host name or requesterLocale. Check the requester configuration.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message.

**User Response:**

- None.

**SOAX2500E:{0}{1} Unsupported operation detected: [{2}].**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the name of an unsupported operation. The service implementation does not support the Web Service operation that was received. Release 1 only supports chargeAmount, refundAmount, chargeVolume, and refundVolume operation requests.

**User Response:**

- Actions are Web Service client specific. If possible, disable the Web Service client's ability to send the unsupported operation requests.

**SOAX2501I:{0}{1} WriteUsageRecordRequest = {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the contents of a WriteUsageRecordRequest. This message contains the contents of the WriteUsageRecordRequest that will be used to write a payment usage record to the USAGERECORD and USAGEPROPERTIES tables.

**User Response:**

- None.

**SOAX2502E:{0}{1} Events client reported fault while writing a change record: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is a general summary of the problem. A payment charge record could not be

written to the CEI repository because of network or database problems. The CEI emitter will throw one of the following exceptions:

- com.ibm.events.DataStoreConnectionFailureException
- com.ibm.events.datastore.BadConnectionException
- com.ibm.events.datastore.ConnectionTimeoutException
- com.ibm.events.datastore.DataStoreNamingException
- com.ibm.events.datastore.DataStoreIoException
- com.ibm.events.datastore.DataStoreSqlException
- javax.ejb.TransactionRolledbackLocalException

If possible, an alarm will be sent to an administrator. This may not be possible if the CEI server is completely down. Message SOAX2601 contains the suggested action that will be sent with the alarm.

**User Response:**

- Verify that the CEI server is correctly configured and deployed.
- Verify that the CEI database is full or is having connectivity problems.

**SOAX2503W:{0}{1} Events client reported fault while writing a change record: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is a general summary of the problem. A payment charge record could not be written to the CEI repository due to problems that are not related to the network or database. If possible, an alarm will be sent to an administrator. This may not be possible if the CEI server is completely down. Message SOAX2601 contains the suggested action that will be sent with the alarm.

**User Response:**

- Verify that the CEI server is correctly configured and deployed.
- Verify that the service implementation modules started without errors.
- Verify that you have correctly configured the service implementation application.
- If you have correctly configured and deployed the CEI server and service implementation, then there may be an internal processing error in the service implementation.

**SOAX2504E:{0}{1} An unexpected exception occurred.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. An unexpected exception occurred in the service implementation that will halt processing of the Web Service request.

**User Response:**

- Verify that the service implementation modules started without errors.
- Verify that you have correctly configured the service implementation application.
- If the service implementation has started without errors and you have correctly configured it, it may be due to an internal error processing the data in the Web Service request.

**SOAX2505W:{0}{1} An unexpected exception occurred. Processing will continue.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. An unexpected exception occurred in the service implementation, but processing of the Web Service request will continue if possible.

**User Response:**

- Verify that the service implmentation modules started without errors.
- Verify that you have correctly configured the service implementation application.
- If the service implementation has started without errors and you have correctly configured it, it may be due to an internal error processing the data in the Web Service request.

**SOAX2506E:{0}{1} Payment records were not created due to policy or configuration settings.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A payment charge record cannot be written to either the USAGERECORDS and USAGEPROPERTIES tables or to the CEI repository because of current configuration settings.

**User Response:**

- Configure one of the following policies to enable writing of a payment charge record: service.config.Payment_PostPaid_IMS.PaymentUsageRecordEnabled, service.config.Payment_PostPaid_IMS.CEIEnabled
- Or configure one of the following payment properties in the Administrative console to enable writing of a payment charge record: PaymentUsageRecordEnabledDefault, CEIEnabledDefault.

**SOAX2507E:{0}{1} Unable to create a CEI event because the Common Events Infrastructure service is not started.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A payment charge record cannot be written to the CEI repository because the Common Event Infrastructure service is not started.

**User Response:**

- Either set the service.config.Payment_PostPaid_IMS.CEIEnabled policy or the Administrative console payment CEIEnabled property to false.
- Or start the Common Event Infrastructure service.

**SOAX2601I:{0}{1} Access to the Common Events Infrastructure datastore is not available. Operator intervention may be required. Verify that the datastore is available.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. This is the suggested action when a payment charge record cannot be written to the CEI repository. It is associated with message SOAX2502.

**User Response:**

- Take the action that is suggested in this message.

**SOAX2602I:{0}{1} Start the Common Events Infrastructure (CEI) service or disable writing payment records to the CEI repository.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. This is the suggested action when a payment charge record cannot be written to the CEI repository. It is associated with message SOAX2507.

**User Response:**

- Take the action that is suggested in this message.

**SOAX3500I:{0}{1} Unable to parse presence document {2} with Content-Type {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presence document the implementation could not parse *{3}* is the Content-Type reported for the presence document The service implementation could not parse the given presence document. This can occur because the document is empty, malformed, or otherwise incomprehensible to the implementation.

**User Response:**

- Verify that the presence server is returning information in a format known to the service implementation

**SOAX3501W:{0}{1} Error contacting the presence server: {2}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the SIP error message returned from attempting to contact the presence server The service implementation could not contact the configured presence server because no endpoint was configured, because of a request timeout, or because of a general SIP error that prevented communication.

**User Response:**

- Verify that the presence server is answering incoming SIP requests.
- Verify that no firewalls are blocking traffic between the service implementation and the presence server.

**SOAX3502W:{0}{1} Error ″{2}″ substituting variables into message ″{3}″: [{4}{5}{6}{7}{8}{9}{10}{11}{12}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the Java error message received while trying to perform text substitution The service implementation could not substitute message variables into a Parlay X-style message string. This could be because the format string is invalid, there are insufficient or too many variables, or the variables could not be processed.

**User Response:**

- If the message format string and variables are from an external system, verify that the external system is providing valid Parlay X-style messages/variables per 3GPP TS 29.199 v6.3.0 Part 1 Section 10.1.

**SOAX3503I:{0}{1} The PresenceSupplier publish operation is not implemented in this release.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A client attempted to invoke the Parlay X PresenceSubscriber publish operation, which is not implemented in this release.

**User Response:**

- Verify that the client can recover from the resulting ServiceException.

**SOAX3504I:{0}{1} The PresenceSupplier getOpenSubscriptions operation is not implemented in this release.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A client attempted to invoke the Parlay X PresenceSubscriber getOpenSubscriptions operation, which is not implemented in this release.

**User Response:**

- Verify that the client can recover from the resulting ServiceException.

**SOAX3505I:{0}{1} The PresenceSupplier updateSubscriptionAuthorization operation is not implemented in this release.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A client attempted to invoke the Parlay X PresenceSubscriber updateSubscriptionAuthorization operation, which is not implemented in this release.

**User Response:**

- Verify that the client can recover from the resulting ServiceException.

**SOAX3506I:{0}{1} The PresenceSupplier getMyWatchers operation is not implemented in this release.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A client attempted to invoke the Parlay X PresenceSubscriber getMyWatchers operation, which is not implemented in this release.

**User Response:**

- Verify that the client can recover from the resulting ServiceException.

**SOAX3507I:{0}{1} The PresenceSupplier getSubscribedAttributes operation is not implemented in this release.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A client attempted to invoke the Parlay X PresenceSubscriber getSubscribedAttributes operation, which is not implemented in this release.

**User Response:**

- Verify that the client can recover from the resulting ServiceException.

**SOAX3508I:{0}{1} The PresenceSupplier blockSubscription operation is not implemented in this release.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. A client attempted to invoke the Parlay X PresenceSubscriber blockSubscription operation, which is not implemented in this release.

**User Response:**

- Verify that the client can recover from the resulting ServiceException.

**SOAX3509I:{0}{1} Unable to retrieve presence information for presentity {2}. Service received SIP response {3} to an outbound SUBSCRIBE request with Call-ID {4}.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity whose presence could not be retrieved. *{3}* is the SIP response code received for the outbound SUBSCRIBE request. *{4}* is the Call-ID used in the outbound SUBSCRIBE request. The service implementation could not retrieve presence information for the given presentity. The implementation received the SIP response code for an outbound SUBSCRIBE request on the given Call-ID.

**User Response:**

- If the response code is a 3xx redirect response, verify that the presence server URI is correctly configured and does not point to a SIP redirect server. The service implementation will not follow 3xx redirect responses.
- If the response code is a 408 Request Timeout, verify that the configured presence server URI is correct and that the presence server is executing.
- Resolve other response codes based upon their meanings in RFCs 3265 and 3261. The Call-ID may be useful for locating specific messages within both the product and the presence server logs.

**SOAX3510I:{0}{1} Unable to retrieve presence information for presentity {2} within {3} millisecond(s).**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity whose presence could not be retrieved. *{3}* is the SIP fetch timeout in milliseconds used to retrieve the information. The service implementation could not retrieve presence information for the given presentity within the given timeout period.

**User Response:**

- Ensure that the presence server is able to respond to an incoming SUBSCRIBE request and generate an outbound NOTIFY within the given timeout.
- Ensure that the network between the service and the presence server allows for three SIP messages (example: SUBSCRIBE/200 then NOTIFY) within the given time period.
- Ensure that the presence server and the network can meet the above two conditions while under load. If necessary, adjust the traffic shaping settings for the presence server to enforce a maximum load.

**SOAX3511I:{0}{1} Unable to retrieve presence information for target {2}. The URI scheme was not present within the AdditionalGroupURISchemes setting, and the service was unable to parse target as a SIP, SIPS, or TEL URI: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the target whose presence could not be retrieved. *{3}* is exception message received when attempting to parse the given target. The service implementation could not retrieve presence information for the given target because the target could not be converted into a SIP, SIPS, or TEL URI. The URI scheme was not present within the AdditionalGroupURISchemes setting.

**User Response:**

- Ensure that the target follows from the supplied request address. This may require verifying group URI mappings within the group list resolution service in use.
- Ensure that the AdditionalGroupURISchemes setting is correct, and that it contains the groupScheme setting in use at the Group List Manager.
- Ensure that the target is a valid SIP, SIPS, or TEL URI.

**SOAX3512I:{0}{1} Requested notification duration, {2} milliseconds, is longer than maximum allowable duration, {3} milliseconds.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the requested notification duration in milliseconds. *{3}* is the maximum allowable notification duration in milliseconds. The client's request was rejected because it specified too long a notification duration.

**User Response:**

- Ensure that the service's MaximumNotificationDuration setting is appropriate. Increase the setting if desired.
- Ensure that incoming policy headers that modify the MaximumNotificationDuration are appropriate. Increase the policy value if desired.
- Have the Parlay X client request a shorter duration.

**SOAX3517I:{0}{1} Unable to augment existing subscription for presentity {2} with endpoint {3} and correlator "{4}". The newly requested correlator "{5}" differs from the previous correlator.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity to which the subscription applies. *{3}* is the PresenceNotification endpoint the client specified when originally creating the subscription. *{4}* is the correlator the client specified when originally creating the subscription. *{5}* is the correlator the client specified in the rejected request. The client's subscribePresence request was rejected because it the client already has a valid subscription for the given presentity with a different correlator. Clients must reuse any previously specified correlator and endpoint when issuing a second subscribePresence request for a given presentity.

**User Response:**

- Ensure that the client re-uses any previously specified correlator and endpoint. Once you make the appropriate changes to the correlator and endpoint, the client should retry the request, otherwise the subscription permissions may be in a consistent state.
- If you cannot use the previous correlator, use a new session when making this request. Using a new session avoids this request colliding with any state from earlier requests.

**SOAX3518I:{0}{1} Unable to augment existing subscription for presentity {2} with endpoint {3} and correlator ″{4}″. The newly requested endpoint {5} differs from the previous endpoint.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity to which the subscription applies. *{3}* is the PresenceNotification endpoint the client specified when originally creating the subscription. *{4}* is the correlator the client specified when originally creating the subscription. *{5}* is the correlator the client specified in the rejected request. The client's subscribePresence request was rejected because it the client already has a valid subscription for the given presentity with a different endpoint. Clients must re-use any previously specified correlator and endpoint when issuing a second subscribePresence request for a given presentity.

**User Response:**

- Ensure that the client reuses any previously specified correlator and endpoint. Once the appropriate changes are made to the correlator and endpoint, the client should retry the same request. Otherwise the subscription permissions may be in an consistent state.
- If the previous correlator cannot be reused, use a new session when making this request. Using a new session avoids this request colliding with any state from earlier requests.

**SOAX3519I:{0}{1} Unable to retrieve presence information for presentity {2}. Service generated SIP response {3} to an inbound NOTIFY request with Call-ID {4}.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity whose presence could not be retrieved. *{3}* is the SIP response sent for the inbound NOTIFY request. *{4}* is the Call-ID of the inbound NOTIFY request. The service implementation could not retrieve presence information for the given presentity. The service implementation received a malformed or otherwise invalid NOTIFY request on the given Call-ID. The service sent a SIP response with the given error code to the presence server because of the inbound NOTIFY.

**User Response:**

- If the response has a status of 400, verify that the incoming NOTIFY is well-formed according to RFC 3265. Among other things, a well-formed NOTIFY request contains both an Event and a Subscription-State header.
- If the response has a status of 415, verify that the incoming NOTIFY has a Content-Type compatible with the Accept headers in the outbound SUBSCRIBE.

- If the response has a status of 489, verify that the incoming NOTIFY has an Event header containing the same SIP event package as the one specified in the corresponding outbound SUBSCRIBE.
- Resolve other response codes based upon their meanings in RFCs 3265 and 3261. The Call-ID may be useful for locating specific messages within both the product and the presence server logs.

**SOAX3520I:{0}{1} Unable to retrieve presence information for presentity {2}. Service received an unrecoverable Subscription-State of {3} in an inbound NOTIFY request with Call-ID {4}.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity whose presence could not be retrieved. *{3}* is the Subscription-State header from the inbound NOTIFY request. *{4}* is the Call-ID of the inbound NOTIFY request. The service implementation could not either retrieve or re-subscribe to presence information for the given presentity. The service implementation received a Subscription-State header on an inbound NOTIFY request indicating that further subscription attempts were disallowed according to the semantics of RFC 3265.

**User Response:**

- If the Subscription-State header has a ’reason=rejected’ parameter, the subscription has been terminated due to a change in authorization policy. The service implementation’s behavior follows from RFC 3265 section 3.2.4. If the behavior is unacceptable, determine why the presence server reported a change in authorization policy.
- If the Subscription-State header has a ’reason=noresource’ parameter, the subscription has been terminated because the presentity which was being monitored no longer exists. The service implementation’s behavior follows from RFC 3265 section 3.2.4. If the behavior is unacceptable, determine why the presence server reported that the presentity ceased to exist.
- Resolve other reason parameters based upon their meanings in RFC 3265 The Call-ID may be useful for locating specific messages within both the product and the presence server logs.

**SOAX3521I:{0}{1} Unable to retrieve presence information for presentity {2}. Service cannot process a presentity with a TEL URI when the PresenceServerSIPURI setting is empty.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the presentity whose presence could not be retrieved. The service implementation cannot retrieve information for the given presentity because it has a TEL URI. When the PresenceServerSIPURI setting is empty, the service implementation relies on the presentity URI to direct an outbound SUBSCRIBE request to the appropriate destination. The SIP Container cannot determine where to send a TEL URI without additional routing information in the form of a non-empty PresenceServerSIPURI.

**User Response:**

- Ensure that the given presentity should be a TEL URI.
- Configure a SIP- or SIPS-based PresenceServerSIPURI that understands how to either respond to or direct TEL URIs.

**SOAX3522E:{0}{1} The configured PresenceServerSIPURI {2} is not a valid SIP or SIPS URI: {3}**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the configured PresenceServerSIPURI. *{3}* is the exception thrown while attempting to parse the URI. At runtime, the service cannot parse the supplied PresenceServerSIPURI into a valid SIP or SIPS URI. It cannot service any requests associated with this setting.

**User Response:**

- Modify the PresenceServerSIPURI setting to contain a valid, well-formed SIP or SIPS URI.

**SOAX3531I:{0}{1} Unable to retrieve members for group URI {2}. Incoming SOAP headers lacked relevant twssHeaders/resolvedGroups/groupList content.**

**Explanation:**

*{0}* is the transaction ID for the Parlay X Web Service request. *{1}* is the name of the service implementation that issued the message. *{2}* is the incoming group URI from the Parlay X Web Service request. The service could not obtain information about the given group URI from the SOAP headers associated with the Parlay X request. The service expects that the SOAP headers contain an entry every incoming group URI, and without the header information the service cannot process the request.

**User Response:**

- Ensure that the group URI is as intended.
- Ensure that the group URI is defined with the Group List Manager.
- Ensure that the ESB is installed and processing incoming requests using a flow containing the Group Resolution Mediation primitive.
- Ensure that the Group Resolution Mediation primitive is successfully contacting the Group List Manager

## Messages_en_US

This section documents the Parlay X fault messages.

**A policy error occurred. Error code is %1.**

**Explanation:**

A policy error occurred. Error code is %1. This means that the client application has exceeding some usage restrictions.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Privacy verification failed for address %1, request is refused.**

**Explanation:**

Privacy verification failed for address %1. Request is refused. This means that the client application is trying to access a user device that it is not authorized for.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Too many addresses specified in message part %1.**

**Explanation:**

Too many addresses specified in message part %1. The client application has passed in too many addresses for the operation.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Unlimited notification request not supported.**

**Explanation:**

Unlimited notification request not supported. The client application should limit the notification requests.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Too many notifications requested.**

**Explanation:**

Too many notifications requested. The client application should not request as many notifications.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Group specified in message part %1 not allowed.**

**Explanation:**

Group specified in message part %1 not allowed. Groups are limited to certain operations.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Nested group specified in message part %1 not allowed.**

**Explanation:**

Nested group specified in message part %1 not allowed.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Charging is not supported.**

**Explanation:**

Charging is not supported. Charging is supported on only some operations.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Invalid frequency requested.**

**Explanation:**

Invalid frequency requested.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Busy criteria is not supported.**

**Explanation:**

Busy criteria is not supported.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Attempt to exceed maximum number of members in a group. Maximum number allowed is %1.**

**Explanation:**

Attempt to exceed maximum number of members in a group. Maximum number allowed is %1.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Attempted to add a group to an existing group. Subgroups are not supported.**

**Explanation:**

Attempted to add a group to an existing group. Subgroups are not supported. The service has some limitations on group management.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Group name is too long. Maximum length allowed is %1.**

**Explanation:**

Group name is too long. Maximum length allowed is %1. The group database has a defined maximum length.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Group URI %1 already exists. Group not created.**

**Explanation:**

Group URI %1 already exists. Group not created.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Vouchers not accepted.**

**Explanation:**

Vouchers not accepted.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Requested accuracy is not supported.**

**Explanation:**

Requested accuracy is not supported.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Geographic notification is not available**

**Explanation:**

Geographic notification is not available.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Periodic notification is not available**

**Explanation:**

Periodic notification is not available.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Too many participants**

**Explanation:**

Too many participants. A limited number is supported by this operation.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Unavailable media.**

**Explanation:**

Unavailable media. The referenced media is not available.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Maximum duration exceeded. Maximum allowed is %1 seconds.**

**Explanation:**

Maximum duration exceeded. Maximum allowed is %1 seconds.

**User Response:**

- Refer to the Parlay X specification for specific details.

**A service error occurred. Error code is %1.**

**Explanation:**

A service error occurred. Error code is %1.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Invalid input value for message part %1.**

**Explanation:**

Invalid input value for message part %1.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Invalid input value for message part %1, valid values are %2.**

**Explanation:**

Invalid input value for message part %1, valid values are %2.

**User Response:**

- Refer to the Parlay X specification for specific details.

**No valid addresses provided in message part %1**

**Explanation:**

No valid addresses provided in message part %1.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Correlator %1 specified in message part %2 is a duplicate.**

**Explanation:**

Correlator %1 specified in message part %2 is a duplicate.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Group %1 in message part %2 is not a valid group.**

**Explanation:**

Group %1 in message part %2 is not a valid group.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Invalid charging information.**

**Explanation:**

Invalid charging information.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Overlapped Criteria %1.**

**Explanation:**

Overlapped Criteria %1. Multiple notification requests have been made for the same device, and only one is supported.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Accuracy of location is not within acceptable limit.**

**Explanation:**

Accuracy of location is not within acceptable limit.

**User Response:**

- Refer to the Parlay X specification for specific details.

**No subscription request from watcher %1 for attribute %2**

**Explanation:**

No subscription request from watcher %1 for attribute %2.

**User Response:**

- Refer to the Parlay X specification for specific details.

**%1 is not a watcher**

**Explanation:**

%1 is not a watcher.

**User Response:**

- Refer to the Parlay X specification for specific details.

**End user authentication failed.**

**Explanation:**

End user authentication failed.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Voucher %1 is not valid.**

**Explanation:**

Voucher %1 is not valid.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Call has already been connected, it cannot be cancelled.**

**Explanation:**

Call has already been connected; it cannot be cancelled.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Call has already been terminated.**

**Explanation:**

Call has already been terminated.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Charging operation failed, the charge was not applied.**

**Explanation:**

Charging operation failed; the charge was not applied.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Message too long. Maximum length is %1 characters.**

**Explanation:**

Message too long. Maximum length is %1 characters.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Data format not recognized for message part %1.**

**Explanation:**

Data format not recognized for message part %1. The client application should use a supported format.

**User Response:**

- Refer to the Parlay X specification for specific details.

**Delivery Receipt Notification not supported.**

**Explanation:**

Delivery Receipt Notification not supported. The client application will not know if the message is successfully received by the target device, only that it is successfully sent.

**User Response:**

- Refer to the Parlay X specification for specific details.

## Messages_en_US

This pseudo class documents the set of messages that can be issued by the IBM Telecom Web Services Server components.

**TAS5001E: An unexpected error occurred.**

**Explanation:**

An unexpected error occurred. The exception is printed and a stack trace is generated.

**User Response:**

Use the exception to determine the type of error, and the stack trace to determine where the error occurred.

**TAS5002E: Attempt to sign Service Agreement for {0} was not successful.**

**Explanation:**

*{0}* is the name of the requested service manager. The ParlayConnector was not successful signing the service agreement for the requested service manager.

**User Response:**

- Verify that the service agreement is correctly configured. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation

**TAS5003E: Requested service {0} was not found.**

**Explanation:**

*{0}* is the name of the requested service manager. The requested service was not found in the list of subscribed services for the caller's domain on the Gateway.

**User Response:**

- Verify that you are using the appropriate service manager factory to obtain the requested service manager.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation

**TAS5004E: Not able to get IpAccess from ParlayConnector.**

**Explanation:**

The ParlayConnector was unsuccessful getting access to the Gateway. See the preceding messages for more information about the failure.

**User Response:**

- Examine the preceding messages for more details about the failure. Correct the problems described in those messages.
- Retry the operation

**TAS5005E: The specified encryption method {0} requires an encryption key.**

**Explanation:**

*{0}* is the name of the encryption method. The encryption method is specified in the configuration, but no encryption key is specified for it.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5006W: The specified encryption method {0} is not supported in this method.**

**Explanation:**

*{0}* is the name of the encryption method. The encryption method is specified in the configuration, but is not supported in the current release.

**User Response:**

Correct the configuration. For information about supported encryption methods, see the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5007W: Property {0} does not have the form serviceType,propertyName,valueList .**

**Explanation:**

*{0}* is a configuration property. The name specifies a service property but the value does not have the required format.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5008E: A Gateway Exception was thrown during IpAccess.obtainInterface({0}).**

**Explanation:**

*{0}* is the name of the Parlay interface which could not be obtained. An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5009E: A Gateway Exception was thrown during retry IpAccess.obtainInterface({0}).**

**Explanation:**

*{0}* is the name of the Parlay interface which could not be obtained. An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5010I: Retrying IpAccess.obtainInterface().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5011E: A Gateway Exception was thrown during IpServiceDiscovery.listSubscribedServices().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5012E: A Gateway Exception was thrown during retry IpServiceDiscovery.listSubscribedServices().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5013I: Retrying IpServiceDiscovery.listSubscribedServices().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5014E: A Gateway Exception was thrown during IpServiceAgreementManagement.selectService().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5015E: A Gateway Exception was thrown during retry IpServiceAgreementManagement.selectService().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5016I: Retrying IpServiceAgreementManagement.selectService().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5017E: A Gateway Exception was thrown during IpServiceDiscovery.listServiceTypes().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5018E: A Gateway Exception was thrown during retry IpServiceDiscovery.listServiceTypes().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5019I: Retrying IpServiceDiscovery.listServiceTypes().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5020E: A Gateway Exception was thrown during IpServiceAgreementManagement.signServiceAgreement().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5021E: A Gateway Exception was thrown during retry IpServiceAgreementManagement.signServiceAgreement().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5022I: Retrying IpServiceAgreementManagement.signServiceAgreement().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5023W: Not able to get Gateway IpServiceDiscovery object.**

**Explanation:**

The Gateway returned a NULL Service Discovery object reference.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation

**TAS5024W: No services are available from the Gateway.**

**Explanation:**

The Gateway returned an empty Subscribed Services list.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation

**TAS5025W: The requested service {0} was not found in the list of services available on the Gateway.**

**Explanation:**

*{0}* is the type of service manager requested. The specified service was not found in the Subscribed Services list returned by the Gateway.

**User Response:**

- Verify that you are using the appropriate service manager factory to obtain the requested service manager.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation

**TAS5026W: The requested service {0} was found in list of services available on the Gateway, but the corresponding service token is null.**

**Explanation:**

*{0}* is the type of service manager requested. The specified service was found in the Subscribed Services list returned by the Gateway, but the service token returned by the Gateway is NULL.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation

**TAS5027W: The number of service types is greater than {0}; only the first {1} will be listed.**

**Explanation:**

*{1}* is the maximum number of service types that can be listed. *{2}* is the actual number of service types that will be listed. The Subscribed Services list returned by the Gateway contains more entries than expected. Not all service types in the Gateway Subscribed Services list will be reported.

**User Response:**

None.

**TAS5028W: Not able to get Gateway signatureAndServiceManager object.**

**Explanation:**

The ParlayConnector was not successful in signing the Service Agreement for the requested service. See the preceding messages for more information about the failure.

**User Response:**

- Examine the preceding messages for more details about the failure. Correct the problems described in those messages.
- Retry the operation

**TAS5029W: Property {0} was not found in the configuration.**

**Explanation:**

*{0}* is a configuration property. The named property is not defined in the configuration. If a default value is defined for the property, the default value will be used.

**User Response:**

If you intended to configure the named property, correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5030W: Property {0}, defined in the configuration, has an unrecognized value. Explanation:**

*{0}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. If a default value is defined for the property, the default value will be used.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5031W: No service agreement text was found in the configuration. Explanation:**

The ParlayConnector configuration specifies that service agreements are defined by service type, but no service agreements are configured.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5032W: Property {0}, defined in the configuration, will not be used for service discovery.**
**Explanation:**

*{0}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. The ParlayConnector will ignore this property.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5033E: Property {0}, defined in the configuration, is incorrect. It must be greater than zero, but not greater than one.**
**Explanation:**

*{0}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. This value should be greater than 0.0, but not greater than 1.0.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5034E: A P_SERVICE_ACCESS_DENIED exception on selectService() for service type {0} may indicate that there are service agreements that are not terminated. If the problem persists, contact your Gateway provider.**
**Explanation:**

*{0}* identifies a service type registered on the Parlay Gateway. A P_SERVICE_ACCESS_DENIED exception was thrown by the Parlay Gateway IpServiceAgreementManagement.selectService() for the specified service type. The most likely reasons are:

- The client domain is not subscribed to the requested service.

- A service agreement has already been signed for the requested service, but ParlayConnector is unable to terminate this agreement.

**User Response:**

- Verify that the client domain is configured correctly. If it is not, correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Verify that the client domain that is configured is subscribed to the requested service. If it is not, contact your Gateway provider or redesign your application to use a different service type.
- If the problem persists, contact your Gateway provider. If there is a signed, unterminated service agreement for this service type, request that the service agreement be terminated.

**TAS5035E: SERVICE{0}_IDENTIFIER {1} is configured, but SERVICE{2}_MAPPEDCLASS is not configured.**

**Explanation:**

*{0}* identifies a Gateway extension service is configured in the environment. *{2}* is the Gateway identifier string configured for the Gateway extension service. *{3}* identifies the mapped class which must be configured for the Gateway extension service. The specified Gateway extension service is configured with a Gateway identifier string, but the required corresponding mapped class is not configured.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5036E: Extension Service {0} will not be enabled.**

**Explanation:**

*{0}* is the Gateway identifier string configured for a Gateway extension service. The specified Gateway extension service, as currently configured, cannot be supported. See the preceding messages for more information about the problem.

**User Response:**

- Examine the preceding messages for more information about the problem.
- Correct the problems described in those messages.
- Restart the Application.
- Retry the operation

**TAS5037E: SERVICE{0}_MAPPEDCLASS is configured as {1}, but a classname cannot end with the path delimiter character.**

**Explanation:**

*{1}* identifies a Gateway extension service mapped class configured in the environment. *{2}* is the class name configured for the Gateway extension service mapped class. The specified Gateway extension service mapped class is not a properly formed class name because it ends with the path delimiter character.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5038E: SERVICE{0}_MAPPEDCLASS is configured as {1}; not able to load required associated class {2}.**

**Explanation:**

*{1}* identifies a Gateway extension service mapped class configured in the environment. *{2}* is the class name configured for the Gateway extension service mapped class. *{3}* identifies a class which was not found. The specified associated class must exist for the specified mapped class to be a properly formed mapped class.

**User Response:**

Correct the configuration or the mapped class name. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5039E: Not able to map the SERVICE{0}_IDENTIFIER {1} service manager to the SERVICE{2}_MAPPEDCLASS {3}.**

**Explanation:**

*{1}* identifies a Gateway extension service configured in the environment. *{2}* is the Gateway identifier string configured for the Gateway extension service. *{3}* identifies the mapped class configured for the Gateway extension service. *{4}* is the class name configured for the Gateway extension service mapped class. The specified Gateway extension service cannot be mapped to the mapped class configured for it. See the preceding messages for more information about the problem.

**User Response:**

- Examine the preceding messages for more details about the problem. Correct the problems described in those messages.
- Retry the operation

**TAS5040W: Property {0} has the form serviceType,propertyName,and valueList, but valueList is null.**

**Explanation:**

*{0}* is a configuration property. The named property specifies a service property in the correct format, but all entries in the valueList are NULL.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5041E: Not able to map the requested service manager {0}.**

**Explanation:**

*{0}* is the type of service manager requested. The ParlayConnector was able to get the specified service manager from the Gateway, but was unable to create the associated mapped class object. See the preceding messages for more information about the problem.

**User Response:**

- Examine the preceding messages for more details about the problem. Correct the problems described in those messages.
- Retry the operation

**TAS5042E: SERVICE{0}_MAPPEDCLASS ({1}) is not a properly formed class name. Explanation:**

*{1}* identifies a Gateway extension service mapped class configured in the environment. *{2}* is the class name configured for the Gateway extension service mapped class. The specified Gateway extension service mapped class is not a properly formed class name.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5043E: Class {0} for SERVICE{1}_MAPPEDCLASS {2} does not contain the required narrow() method.**

**Explanation:**

*{1}* identifies a required class associated with the specified mapped class. *{2}* identifies the mapped class configured for the Gateway extension service. *{3}* is the class name configured for the Gateway extension service mapped class. The specified required class was found, but it is not properly formed because it does not contain a narrow() method.

**User Response:**

- Correct the required class definition.
- Rebuild the Application.
- Restart the Application.
- Retry the operation.

**TAS5044E: SERVICE{0}_MAPPEDCLASS is configured as {1}. Not able to load required associated interface {2}.**

**Explanation:**

*{1}* identifies a Gateway extension service mapped class configured in the environment. *{2}* is the class name configured for the Gateway extension service mapped class. *{3}* identifies the interface that was not found. The specified associated interface must exist for the specified mapped class to be a properly formed mapped class.

**User Response:**

Correct the configuration or the mapped class. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5045E: Class {0} for SERVICE{1}_MAPPEDCLASS {2} does not contain the required {3}({4}) method.**

**Explanation:**

*{1}* identifies a required class associated with the specified mapped class. *{2}* identifies the mapped class configured for the Gateway extension service. *{3}* is the class name configured for the Gateway extension service mapped class. *{4}* is the name of the method that was not found in class *{1}*. *{5}* is the class of the single parameter required for method *{4}*. The specified required class was found, but it is not properly formed because it does not contain a method with the specified signature.

**User Response:**

- Correct the required class definition.
- Rebuild the Application.
- Restart the Application.
- Retry the operation.

**TAS5046E: A PLATFORM_EXCEPTION on terminateServiceAgreement() service type {0} might cause an unterminated service agreement.**

**Explanation:**

*{0}* identifies a service type registered on the Parlay Gateway. A PLATFORM_EXCEPTION was thrown when attempting to call IpServiceAgreementManagement.terminateServiceAgreement() for the specified service type. The most likely reasons are:

- A null parameter was passed to IpServiceAgreementManagement.terminateServiceAgreement().
- A hardware communication failure.

**User Response:**

Check the console for more specific information about the problem. If there was a communication failure, fix the problem.

- Stop the IBM Telecom Web Services Server.
- Restart the IBM Telecom Web Services Server. The ParlayConnector will retry the IpServiceAgreementManagement.terminateServiceAgreement() operation.

**TAS5047W: Property {0} does not have the form serviceType,agreementText .**

**Explanation:**

*{0}* is a configuration property. The named property specifies agreement text for a service agreement, but the value does not have the required format.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5048W: Not able to get Gateway IpServiceAgreementManagement object.**

**Explanation:**

The ParlayConnector was not able to obtain the IpServiceAgreementManagement interface from the ParlayGateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5049I: Service manager for {0} is preloaded.**

**Explanation:**

*{0}* is a service type supported by the Gateway. The ParlayConnector has obtained the specified service manager using the configured service agreement and has cached the service manager reference.

**User Response:**

None.

**TAS5050I: The current state of {0} is {1}.**

**Explanation:**

*{1}* identifies an application server. *{2}* is a component state. The specified component is reporting its current state.

**User Response:**

None.

**TAS5051E: No IpService was specified to isWireless().**

**Explanation:**

The parameter specified to the isWireless() method is not a service manager object.

**User Response:**

- Correct the application to specify a service manager object to the isWireless() method.
- Rebuild the Application.
- Restart the Application.
- Retry the operation.

**TAS5052E: A Gateway Exception was thrown during IpAccess.obtainInterfaceWithCallback({0}).**

**Explanation:**

*{0}* is the interface that the ParlayConnector was requesting from the Gateway. An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5053E: A Gateway Exception was thrown during retry IpAccess.obtainInterfaceWithCallback({0}).**

**Explanation:**

*{0}* is the interface that the ParlayConnector was requesting from the Gateway. An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5054I: Retrying IpAccess.obtainInterfaceWithCallback().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5055E: A Gateway Exception was thrown during IpServiceDiscovery.describeServiceType().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5056E: A Gateway Exception was thrown during retry IpServiceDiscovery.describeServiceType().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5057I: Retrying IpServiceDiscovery.describeServiceType().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5058I: The ParlayConnector is using the default value {0} for Property {1}.**

**Explanation:**

*{1}* is the default value for the named property. *{2}* is the name of the property. The ParlayConnector is using the default value for the named property.

**User Response:**

If the default value is correct for the application, no action is necessary. If the default value needs to be overridden, correct the configuration. For information about supported encryption methods, see the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5059W: Property {0}, defined in the configuration, has an incorrectly formed value.**

**Explanation:**

*{0}* is a property. The named property is defined in the configuration, but the value assigned to it is not correct. If a default value is defined for the property, the default value will be used.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5060E: A Gateway Exception was thrown during IpServiceAgreementManagement.initiateSignServiceAgreement().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5061E: A Gateway Exception was thrown during retry IpServiceAgreementManagement.initiateSignServiceAgreement().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.

- Correct any existing problems.
- Retry the operation.

**TAS5062I: Retrying IpServiceAgreementManagement.initiateSignServiceAgreement().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5063E: Attempt to initiate sign Service Agreement for {0} was not successful.**

**Explanation:**

*{0}* is the service manager the ParlayConnector was preparing to request from the Gateway. An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5064E: Not able to sign Service Agreement for {0} because the required property {1} was not found in the configuration.**

**Explanation:**

*{1}* is the service manager the ParlayConnector was preparing to request from the Gateway. *{2}* is the name of a required environment property. The ParlayConnector is not able to sign a service agreement to obtain the named service manager from the Gateway. An unexpected error occurred at the Gateway.

**User Response:**

Correct the configuration. For information about supported encryption methods, see the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5065E: An unexpected error occurred in describeService({0}).**

**Explanation:**

*{0}* is the name of the Gateway service specified to describeService(). An unexpected error occurred.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5066W: No properties were found during describeService({0}).**

**Explanation:**

*{0}* is the name of the Gateway service specified to describeService(). No properties were found for the named service, either on the Parlay Gateway, or in the ParlayConnector's environment configuration.

**User Response:**

- Verify the service name that was specified to describeService().
- If necessary, correct the configuration. For information about supported encryption methods, see the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5067W: Not able to verify the digital signature of the Parlay Gateway given with the {0} service manager. The service manager is accepted.**

**Explanation:**

*{0}* is the name of the Gateway service the ParlayConnector has requested from the Parlay Gateway. The ParlayConnector has received the named service manager. However, the ParlayConnector is unable to verify the sender of this object because the ParlayConnector is not configured to verify the digital signature. The service manager is accepted by the ParlayConnector.

**User Response:**

If the communication link to the Gateway is secure, no action is necessary. If the communication link to the Gateway may be vulnerable,

- Do not use the service manager object.
- Configure the ParlayConnector so that the source of the service manager object can be verified. For information about configuring the ParlayConnector, see the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5068W: Property {0}, defined in the configuration has a value that is not recognized {1}. It is not being used.**

**Explanation:**

*{1}* is a property. *{2}* is the incorrect value assigned to the specified property. The named property is defined in the configuration, but the value assigned to it is not correct.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5069E: The Parlay Gateway gave a digital signature that is not correct with the {0} service manager. The service manager is discarded.**

**Explanation:**

*{0}* is the name of the Gateway service the ParlayConnector has requested from the Parlay Gateway. The ParlayConnector has received the named service manager. However, the ParlayConnector cannot verify that the service manager was received from the Parlay Gateway because the digital signature received with it is not correct. The ParlayConnector will not return the service manager to the application or use the service manager in any way.

**User Response:**

- Verify that the ParlayConnector is correctly configured to verify the digital signature received from the Gateway. For information about configuring the ParlayConnector, see the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5070W: Property {0}, defined in the configuration, will not be used for signing service agreements.**

**Explanation:**

*{0}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. The ParlayConnector will ignore this property.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5071W: Property {0}, defined in the configuration, has the form serviceType,agreementText, but serviceType is null, or agreementText is null.**

**Explanation:**

*{0}* is a configuration property. The named property specifies agreement text for a service agreement in the correct format, but the agreement text is NULL.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5072W: Property {0}, defined in the configuration, does not have the form serviceType,terminationText.**

**Explanation:**

*{0}* is a configuration property. The named property specifies termination text for a service agreement, but the value does not have the required format.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5073W: Property {0}, defined in the configuration, has the form serviceType,terminationText, but serviceType is null or terminationText is null.**

**Explanation:**

*{0}* is a configuration property. The named property specifies termination text for a service agreement in the correct format, but the termination text is NULL.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5074W: Property {0}, defined in the configuration, will not be used for terminating service agreements.**

**Explanation:**

*{0}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. The ParlayConnector will ignore this variable.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5075E: An Exception occurred requesting cache status update on target node: {0}.**

**Explanation:**

*{0}* identifies a node in the WebSphere cluster. The ParlayConnector was not able to synchronize the cache on the named node.

**User Response:**

- Check the console for the specific nature of the problem. Correct any existing problems.
- Stop the application server.
- Restart the application server.

**TAS5076E: Not able to update cache status on target node: {0}.**

**Explanation:**

*{0}* identifies a node in the WebSphere cluster. The ParlayConnector was not able to synchronize the cache on the named node.

**User Response:**

- Check the console for more specific information about the problem. Correct any existing problems.
- Stop the application server.
- Restart the application server.

**TAS5077E: An Exception occurred getting a {0} service instance using the configured JNDI name {1}.**

**Explanation:**

*{1}* is the service type name defined for a Parlay service implemented locally. *{2}* is JNDI name specified as the implementation of service type name. The active configuration includes the definition for the specified Parlay service implemented locally. An exception occurred getting an instance of the implementation using the specified JNDI name.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5078E: The {0} service instance obtained using configured JNDI name {1} does not implement the required interface {2}.**

**Explanation:**

*{1}* is the service type name defined for a Parlay service implemented locally. *{2}* is JNDI name specified as the implementation of service type name. *{3}* is an interface that must be implemented by all local implementations of Parlay services defined in the active configuration. The active configuration includes the definition for the specified Parlay service implemented locally. The object obtained using the specified JNDI name does not implement the specified interface.

**User Response:**

Do one of the following:

- Correct the configuration.

- Change the local implementation so that it does implement the required interface.

See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5079E: The {0} service instance obtained using configured JNDI name {1} does not extend the required class {2}.**

**Explanation:**

*{1}* is the service type name defined for a Parlay service implemented locally. *{2}* is JNDI name specified as the implementation of service type name. *{3}* is an Class that must be extended by all local implementations of Parlay services defined in the active configuration. The active configuration includes the definition for the specified Parlay service implemented locally. The object obtained using the specified JNDI name does not extend the specified Class.

**User Response:**

Do one of the following:
- Correct the configuration.
- Change the local implementation so that it does extend the required Class.

See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5080E: Requested service {0} is not type {1}.**

**Explanation:**

*{1}* is a Gateway service identifier. *{2}* is a Gateway service type. A compound name of the form service-type:service-identifier was specified for a Gateway service. The service identifier has been found in the Gateway's list of subscribed services, but the service type in the description does not match the service type in the compound name.

**User Response:**

Verify that the service type and service identifier are correctly specified. If the compound name is incorrect,
- Correct the compound name in the request
- Retry the operation

If the compound name is correct,
- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5081I: The {0} service is enabled.**

**Explanation:**

*{0}* is a Gateway service type. The ParlayConnector has obtained the Gateway reference to the named service.

**User Response:**

None.

**TAS5082E: The ParlayConnector timed out waiting for a Gateway callback in authentication.**

**Explanation:**

The ParlayConnector is not able to complete authentication with the Parlay Gateway because the Gateway has not made the expected call back.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- ParlayConnector will retry authentication when it next receives a service manager request.

**TAS5083E: The ParlayConnector timed out waiting for a Gateway callback to sign the service agreement for the {0} service.**

**Explanation:**

*{0}* is a Gateway service type. The ParlayConnector is not able to sign a service agreement with the Parlay Gateway because the Gateway did not make the expected call back. The ParlayConnector is not able to obtain the requested service manager.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- ParlayConnector will retry signing the service agreement when it next receives a request for the service manager.

**TAS5084E: Not able to get the initial Parlay Gateway reference from the primary remote host.**

**Explanation:**

The ParlayConnector was not able to connect to the Gateway using the configured primary Gateway initial host.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.

**TAS5085E: Not able to get the initial Parlay Gateway reference from the secondary remote host.**

**Explanation:**

The ParlayConnector was not able to connect to the Gateway using the configured primary Gateway initial host. A secondary Gateway initial host is configured. The ParlayConnector is not able to connect to the Gateway using the configured secondary Gateway initial host.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.

**TAS5086W: Configuration data {0} not found for secondary remote host.**

**Explanation:**

*{0}* is a configuration property which is not configured. The ParlayConnector was not able to connect to the Gateway using the value configured for the primary Gateway initial host. A secondary Gateway initial host is not configured.

**User Response:**

If you configure a secondary Gateway initial host in the environment, the ParlayConnector will attempt to connect to this Gateway and use it when there is a failure to connect using the configured primary Gateway initial host. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5087I: The Gateway has issued IpClientAccess.terminateAccess with reason {0}.**

**Explanation:**

*{0}* is the explanation text provided by the Gateway. The application server is shutting down because the Gateway has terminated access to it. The ParlayConnector must reauthenticate with the Gateway and sign service agreements. All applications must renew their notification requests.

**User Response:**

Restart the application server.

**TAS5088I: The initial Parlay Gateway reference has been obtained.**

**Explanation:**

The application server has obtained the IpInitial object for the Parlay Gateway.

**User Response:**

None.

**TAS5089E: An unexpected error occurred in describeService({0},{1}).**

**Explanation:**

*{1}* is the service type name defined for a Parlay service implemented locally. *{2}* is JNDI name specified as the implementation of service type name. The active configuration includes the definition for the specified Parlay service that is implemented locally. An exception occurred calling describeService() on the object bound to the JNDI name.

**User Response:**

Do one of the following:

- Correct the configuration.
- Correct the local implementation so that it does not throw an exception in describeService().

See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5090W: No properties were found during describeService({0},{1}).**

**Explanation:**

*{1}* is the service type name defined for a Parlay service implemented locally. *{2}* is JNDI name specified as the implementation of service type name. The active configuration includes the definition for the specified

Parlay service implemented locally. No service properties were returned by a call to describeService() on the object bound to the JNDI name.

**User Response:**

None.

**TAS5091W: No gateway service instances were found meeting the requirements for type {0} (gateway ServiceTypeName {1}).**

**Explanation:**

*{1}* is the service type specified on the getServiceManager() request. *{2}* is the ServiceTypeName defined on the Parlay gateway.

- If {1} is not the same as {2}, then the active configuration includes the definition of service type alias {1} for gateway ServiceTypeName {2}.
- If the active configuration includes service properties for type {1}, then they were specified in service discovery.
- The gateway did not return any service instances during service discovery for gateway ServiceTypeName {2} (using any service properties specified in the active configuration for type {1}).

**User Response:**

- Check the ServiceTypeName ({2}) to determine if it is a valid subscribed service name.
- Check the active configuration to verify that any service properties specified for service type {1} are correct.
- Check the gateway to determine if there is an active service manager instance of ServiceTypeName {2} and service properties as specified in the configuration for service type {1}.
- Retry the operation.

**TAS5401E: An unexpected error occurred.**

**Explanation:**

An unexpected error occurred. The exception is printed and a stack trace is generated.

**User Response:**

Use the exception to determine the type of error, and the stack trace to determine where the error occurred.

**TAS5402E: A Gateway Exception was thrown during IpAPILevelAuthentication.requestAccess().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5403E: A Gateway Exception was thrown during retry IpAPILeveAuthentication.requestAccess().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.

- Retry the operation.

**TAS5404I: Retrying IpAPILevelAuthentication.requestAccess().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5406I: The ParlayConnector is connected to the gateway.**

**Explanation:**

The ParlayConnector successfully connected with the Gateway.

**User Response:**

None.

**TAS5407W: The ParlayConnector is not connected to the gateway.**

**Explanation:**

The ParlayConnector has not connected with the Gateway.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.

**TAS5408E: A Gateway Exception was thrown during ipInitialRef.initiateAuthentication().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5409E: A Gateway Exception was thrown during retry ipInitialRef.initiateAuthentication().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5410I: Retrying ipInitialRef.initiateAuthentication().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5411E: A Gateway Exception was thrown during IpAPILevelAuthentication.selectEncryptionMethod().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5412E: A Gateway Exception was thrown during retry IpAPILevelAuthentication.selectEncryptionMethod().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5413I: Retrying IpAPILevelAuthentication.selectEncryptionMethod().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5414E: A Gateway Exception was thrown during IpAPILevelAuthentication.authenticate().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5415E: A Gateway Exception was thrown during retry IpAPILevelAuthentication.authenticate().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5416I: Retrying IpAPILevelAuthentication.authenticate().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5420I: The ParlayConnector accepted the Parlay Gateway. Application-Gateway authentication was successful.**

**Explanation:**

The Gateway has been authenticated to the application.

**User Response:**

None.

**TAS5421E: The ParlayConnector rejects the Parlay Gateway. Application-Gateway authentication was not successful.**

**Explanation:**

The Gateway could not be authenticated to the application. The application has halted its attempt to connect with the Gateway.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.

Unless you need to restart the application or server to correct some other problem, it is not necessary to restart it. The ParlayConnector will retry this operation automatically on the next request for a service manager.

**TAS5422E: Configuration data was not found for required property {0}.**

**Explanation:**

*{0}* is a configuration property. The named property is not defined in the configuration. There is no default value for this property, therefore, it must be configured.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5423E: The Parlay Gateway has returned a NULL object for IpAPILevelAuthentication.**

**Explanation:**

The ParlayConnector has established contact with the Gateway, but the Gateway has not returned the object needed for authentication to begin.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5424E: Application encryption capabilities list is empty.**

**Explanation:**

The ParlayConnector makes a list of encryption methods to pass to the Gateway for authentication method negotiation. When the list of encryption methods supported by the ParlayConnector is reduced using configuration data, the list becomes empty. The ParlayConnector cannot continue the authentication process.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5425E: Authentication Method negotiation was not successful.**

**Explanation:**

The ParlayConnector makes a list of encryption methods to pass to the Gateway for authentication method negotiation. This list contains the list of encryption methods supported by the ParlayConnector, reduced using configuration data. The Gateway does not accept any encryption method in the list passed to it by the ParlayConnector, therefore, the authentication process cannot continue.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5426I: Property {0} was not found in the configuration.**

**Explanation:**

*{0}* is a configuration property. The named property is not defined in the configuration. If a default value is defined for the property, the default value will be used.

**User Response:**

If you intended to configure the named property, correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5427W: Property {1}, defined in the configuration, has an unrecognized value {0}.**

**Explanation:**

*{1}* is the incorrect value assigned to the specified property. *{2}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. If a default value is defined for the property, the default value will be used.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5428E: The Parlay Gateway returned a response but did not use the prescribed authentication method.**

**Explanation:**

The Gateway participated in the authentication process, but did not use the encryption method that it chose during authentication method negotiation.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5429E: The Parlay Gateway specified an authentication method is not in the capability list.**

**Explanation:**

The ParlayConnector makes a list of encryption methods to pass to the Gateway for authentication method negotiation. This list contains the list of encryption methods supported by the ParlayConnector, reduced using configuration data. The Gateway participated in the authentication method negotiation, but chose an authentication method not in the list passed to it by the ParlayConnector.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5430W: Encryption method {0}, defined in the configuration, cannot be used because no value is defined for {1}.**

**Explanation:**

*{1}* is an encryption method configured in the environment. *{2}* is a configuration property required for use with the named encryption method, but is not found in the environment. The named encryption method is defined in the configuration, but it cannot be used because the named property is not defined in the configuration.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5431W: Encryption method {0}, defined in the configuration, cannot be used because no values are defined for {1} and {2}.**

**Explanation:**

*{1}* is an encryption method configured in the environment. *{2}* is a configuration property required for use with the named encryption method, but is not found in the environment. *{3}* is a configuration property required for use with the named encryption method, but is not found in the environment. The named encryption method is defined in the configuration, but it cannot be used because the named properties are not defined in the configuration.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5432E: Not able to stop the application server after IpAppAccess.terminateAccess. An application server Force Stop, then Start is required.**

**Explanation:**

The Gateway has terminated the current access session with the application server. The application server must shut down, then reauthenticate with the Gateway and sign service agreements. All applications must renew their notification requests. The application server was not able to shut down automatically.

**User Response:**

Stop the application server. Restart the application server.

**TAS5433E: No callback object provided. IpAppAccess callback object is required.**

**Explanation:**

The caller passed a NULL parameter instead of the required IpAppAccess callback object.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Rebuild and redeploy the application.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5434E: Not able to get IpAPILevelAuthentication object from ParlayGateway.**

**Explanation:**

The ParlayConnector established contact with the Gateway, but the Gateway did not return the object needed for authentication to begin.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5435E: Not able to get IpAccess object from ParlayGateway.**

**Explanation:**

The ParlayConnector established contact with the Gateway, but the initial request for access was not successful.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5436E: Required property {1}, defined in the configuration, has an unrecognized value {0}.**

**Explanation:**

*{1}* is not the correct value assigned to the specified property. *{2}* is a configuration property. The named property is defined in the configuration, but the value assigned to it is not correct. There is no default value for this property, so it must be configured correctly.

**User Response:**

Correct the configuration. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5437E: Not able to locate the Parlay Gateway in the ORB Name Service.**

**Explanation:**

The ParlayConnector is not able to locate the Gateway using the Name Service in the configured ORB.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5438I: IBM WebSphere Telecom Web Services Server @PRODUCT.VERSION.DETAIL@, build level: {0}.**

**Explanation:**

*{0}* identifies the product build level. The ParlayConnector is reporting its product level.

**User Response:**

None.

**TAS5439E: Not able to locate the local ORB Name Service.**

**Explanation:**

The ParlayConnector is not able to locate the Name Service in the local ORB.

**User Response:**

- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5440E: Not able to get the naming context from the local ORB Name Service.**

**Explanation:**

The ParlayConnector is not able to obtain the Naming Context from the local ORB Name Service.

**User Response:**

- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5441E: Not able to get the naming context from the primary remote ORB Name Service.**

**Explanation:**

The ParlayConnector is not able to obtain the Naming Context from the Name Service of the ORB configured as the Gateway Primary Name Service.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5442E: Not able to get the naming context from the secondary remote ORB Name Service.**

**Explanation:**

The ParlayConnector was not able to connect to the Gateway using the ORB configured as the Gateway Primary Name Service. A Gateway Secondary Name Service ORB is also configured. The ParlayConnector is not able to obtain the Naming Context from the Name Service of the ORB configured as the Gateway Secondary Name Service.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5443W: Configuration data {0} was not found for secondary remote ORB Name Service.**

**Explanation:**

*{0}* identifies the property which is not configured. The ParlayConnector was not able to connect to the Gateway using the ORB configured as the Gateway Primary Name Service. A Gateway Secondary Name Service ORB is not configured.

**User Response:**

If you configure a Gateway Secondary Name Service ORB in the environment, the ParlayConnector will attempt to connect to the Gateway using it when there is a failure to connect using the configured Gateway Primary Name Service ORB. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5444E: Not able to find the GatewayAuthenticator class {0}.**

**Explanation:**

*{0}* identifies a Gateway authenticator class name. The ParlayConnector is configured to use the named class for authenticating with the ParlayGateway, but the named class is not found in the classpath.

**User Response:**

- Verify that the class name is correctly specified in the configuration.
- Verify that the class can be found in the classpath.
- Rebuild and redeploy the application.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.

**TAS5445E: Not able to find the GatewayLocator class {0}.**

**Explanation:**

*{0}* identifies a Gateway locator class name. The ParlayConnector is configured to use the named class for locating the ParlayGateway, but it is not found in the classpath.

**User Response:**

- Verify that the class name is correctly specified in the configuration.
- Verify that the class can be found in the classpath.
- Rebuild and redeploy the application.

- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5446E: Not able to create an instance of class {0} using constructor {0}().**

**Explanation:**

*{1}* identifies a class name. *{2}* identifies a class constructor. The named class is required to have a default constructor, but an error occurred when the ParlayConnector attempted to use it.

**User Response:**

- Verify that the class has a default constructor.
- Verify that the correct class is found in the classpath.
- Rebuild and redeploy the application.
- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5447I: The Parlay Gateway accepts the ParlayConnector. Gateway-Application authentication was successful.**

**Explanation:**

The Parlay Gateway has notified the ParlayConnector that the ParlayConnector has successfully completed authenticating itself to the Gateway.

**User Response:**

None.

**TAS5448E: The Parlay Gateway rejects the ParlayConnector. Gateway-Application authentication was not successful.**

**Explanation:**

The Parlay Gateway has notified the ParlayConnector that the ParlayConnector has failed to authenticate itself to the Gateway.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a WebSphere or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.
- Retry the operation.

**TAS5449E: A Gateway Exception was thrown during IpAPILevelAuthentication.authenticationSucceeded().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5450E: A Gateway Exception was thrown during retry IpAPILevelAuthentication.authenticationSucceeded().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5451I: Retrying IpAPILevelAuthentication.authenticationSucceeded().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5452E: A Gateway Exception was thrown during IpAPILevelAuthentication.abortAuthentication().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5453E: A Gateway Exception was thrown during retry IpAPILevelAuthentication.abortAuthentication().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure.
- Correct any existing problems.
- Retry the operation.

**TAS5454I: Retrying IpAPILevelAuthentication.abortAuthentication().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5455E: Not able to get the naming context using the value configured for property {0}.**

**Explanation:**

*{0}* is a configuration property. The ParlayConnector was unable to connect to the Gateway using the value configured for the named property.

**User Response:**

- Verify that the ParlayConnector is correctly configured for the Gateway. See the configuration information in the *IBM WebSphere Telecom Web Services Server information center*.
- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Restart the WebSphere Application Server.

**TAS5456I: Connecting to NameService on host {0}, port {1}.**
**Explanation:**

The ParlayConnector is connecting to the specified hostname and port.

**User Response:**

None.

**TAS5457I: Attempting to resolve the NamingContext for the Gateway.**
**Explanation:**

The ParlayConnector is attempting to resolve the NamingContext for the Gateway. The Parlay Connector tries until the NamingContext is resolved.

**User Response:**

If this message occurs several times, ensure that the gateway location configuration is property bound to the Gateway.

**TAS5458I: Attempt #{0} to resolve the NamingContext for the Gateway.**
**Explanation:**

The ParlayConnector is attempting to resolve the NamingContext for the Gateway. The Parlay Connector tries until the configured retry count is reached or the NamingContext is resolved.

**User Response:**

None

**TAS5459W: The ParlayConnector state is stopped or stopping. The {0} request is not being processed.**
**Explanation:**

*{0}* is a request received by the ParlayConnector. The current ParlayConnector state prevents it from processing the request.

**User Response:**

Retry the operation when the ParlayConnector is in the ’started’ state.

**TAS5460W: The ParlayConnector state is restarting or recovering. The {0} request is not being processed.**
**Explanation:**

*{0}* is a request received by the ParlayConnector. The current ParlayConnector state prevents it from processing the request.

**User Response:**

Retry the operation when the ParlayConnector is in the ’started’ state.

**TAS5461E: More than one serviceToken has been obtained from the Gateway for service type {0}.**
**Explanation:**

*{0}* identifies a Parlay Gateway service type. The ParlayConnector has detected what appears to be a Parlay Gateway error.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Restart the application server.
- Retry the operation.

**TAS5462E: Not able to get IpAccess from ParlayConnector. Requested {0} service manager cannot be obtained.**

**Explanation:**

*{0}* is the name of the requested service manager. The ParlayConnector was not successful getting access to the Gateway, therefore it is not able to obtain the requested service manager. See the preceding messages for more information about the failure.

**User Response:**

- Examine the preceding messages for more details about the failure. Correct the problems described in those messages.
- Check the console to determine if there was a communication, Gateway, or other failure.
- Retry the operation

**TAS5463E: Not able to get Gateway IpServiceAgreementManagement object. Requested {0} service manager cannot be obtained.**

**Explanation:**

*{0}* is the name of the requested service manager. The ParlayConnector was not able to obtain the IpServiceAgreementManagement interface from the ParlayGateway, therefore it is not able to obtain the requested service manager. See the preceding messages for more information about the failure.

**User Response:**

- Examine the preceding messages for more details about the failure. Correct the problems described in those messages.
- Check the console to determine if there was a communication, Gateway, or other failure.
- Retry the operation

**TAS5464W: The ParlayConnector state is faulty. The {0} request is not being processed.**

**Explanation:**

*{0}* identifies a request received by the ParlayConnector. The current ParlayConnector state prevents it from processing the request.

**User Response:**

See **Troubleshooting** in the *IBM WebSphere Telecom Web Services Server information center*.

**TAS5465E: A Gateway Exception was thrown during ipInitialRef.initiateAuthenticationWithVersion().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5466E: A Gateway Exception was thrown during retry ipInitialRef.initiateAuthenticationWithVersion().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5467I: Retrying ipInitialRef.initiateAuthenticationWithVersion().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5468E: A Gateway Exception was thrown during IpAPILevelAuthentication.selectAuthenticationMethod().**

**Explanation:**

An unexpected error occurred at the Gateway. The ParlayConnector will retry the operation.

**User Response:**

None.

**TAS5469E: A Gateway Exception was thrown during retry IpAPILevelAuthentication.selectAuthenticationMethod().**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5470I: Retrying IpAPILevelAuthentication.selectAuthenticationMethod().**

**Explanation:**

The ParlayConnector is retrying the operation that was in progress when an unexpected error occurred at the Gateway.

**User Response:**

None.

**TAS5501E: The connection to a Parlay Gateway framework failed.**

**Explanation:**

An unexpected error occurred at the Gateway.

**User Response:**

- Check the console to determine if there was a communication, Gateway, or other failure. Correct any existing problems.
- Retry the operation.

**TAS5502I: Framework reconnect is not being done, current state of Connector is {0}.**

**Explanation:**

The Parlay Connector is currently being stopped and the framework failover process will not be done.

**User Response:**

- None

**TAS5503I: An invalid Framework Heart Beat timer has expired, and is being removed.**

**Explanation:**

An invalid Heartbeat timer was found from a previous session and will be removed.

**User Response:**

- None

**TAS5504I: Check found a received pulse from Gateway Framework.**

**Explanation:**

A Heartbeat pulse was received from the Parlay Gateway. This is a health check from the Parlay Gateway to the Parlay Connector.

**User Response:**

- None

**TAS5505E: Check did not find a received pulse from Gateway Framework.**

**Explanation:**

A Heartbeat pulse was not received from the Parlay Gateway. The health check from the Parlay Gateway to the Parlay Connector has failed. The framework failover process will be initiated.

**User Response:**

- None

**TAS5506E: Framework recovery will be started.**

**Explanation:**

The framework failover process has been initiated.

**User Response:**

- None

**TAS5507E: Problem with the reference to the Application Manager.**

**Explanation:**

An exception occurred when the Heartbeat Monitor tried to get an identifier for the ApplicationManager EJB for starting the failover process. This identifier is unique for each deployed ApplicationManager EJB.

**User Response:**

Enable trace log and restart the Parlay Connector to get trace information.

**TAS5508I: HeartBeat Check timer started: Pulse time {0}, Missed Count {1}.**

**Explanation:**

A Heartbeat timer has been started with the following values. *{1}* Is the amount of time between each pulse from the Gateway. *{2}* Is the number of pulse opportunities the Gateway is allowed to miss before framework failover processing will be initiated. The timer will check for received pulses from the Gateway using a time value based on the Pulse time multiplied by the number of Missed count.

**User Response:**

- None

**TAS5509E: Framework recovery will attempt recovery without resetting Service information**

**Explanation:**

The framework failover process has been initiated.

**User Response:**

- None

**TAS5510E: Framework recovery has completed**

**Explanation:**

The framework failover process has completed.

**User Response:**

- Check the console to determine the state of the Parlay Connector. Correct any existing problems.

**TAS5511E: Incorrect Framework version parameter**

**TAS5599E: {0}.**

**Explanation:**

Used internally.

**User Response:**

None

**TAS6001E: Unexpected error.**

**Explanation:**

An unexpected error has occurred. An unexpected error is an error that is not normally expected to occur in the system. The error is caught, the user is notified, the exception, if any, is printed, and the system tries to recover. However, normal operation may not resume.

**User Response:**

Please note any information, such as exception information, that is provided with this error. You may need to restart the system.

**TAS6003W: Property {0} in configuration has an unrecognized value.**

**Explanation:**

The value for a Property key in the environment has an unexpected value in it. Where possible defaults are used.

**User Response:**

Correct the value for the given key.

**TAS6004E: Error, could not find {0} in the JNDI environment.**

**Explanation:**

The component is looking for a value key in the JNDI environment. That value is not present. Where possible a default is used.

**User Response:**

Add the key value to the environment configuration information.

**TAS6005E: An Exception occurred while checking the environment.**

**Explanation:**

The component is experienced an error interacting with the JNDI environment. Further information in the form of an Exception should be provided.

**User Response:**

Review the Exception for possible action.

**TAS6006E: An Exception occurred getting state of ManagedObject {0}.**
**Explanation:**

An error occurred while trying to get the provided object's state. Further information in the form of an Exception should be provided.

**User Response:**

Review the Exception for possible action.

**TAS6007E: Incorrect parameters were passed. {0} has a value of {1}.**
**Explanation:**

The method that generated this message was passed an incorrect parameter.

**User Response:**

Review the logs.

**TAS6008E: An unknown ManagementEvent Type {0} was received and ignored.**
**Explanation:**

An unknown ManagementEvent has been received. This Event is logged, and then ignored. The ManagementEvent in question is provided.

**User Response:**

Possible causes for this error are:

- The code base has mixed versions of code in it.
- An incorrect ManagementEvent was generated and passed. System generated ManagementEvents log their origin.

**TAS6009E: AssignmentID is no longer valid. An application server Force Stop, then Start is required.**
**Explanation:**

An error has been found while processing call notifications. The ParlayCallbackManager cannot restore the incorrect call notification.

**User Response:**

1. Use the ParlayWebConsole to force the application server to stop.
2. Use the ParlayWebConsole to start the application server.

**TAS6010E: Notification request cannot be recovered. An application server Force Stop, then Start is required.**
**Explanation:**

An error has been found while processing call notifications. The ParlayCallbackManager cannot restore the incorrect call notification.

**User Response:**

1. Use the ParlayWebConsole to force the application server to stop.
2. Use the ParlayWebConsole to start the application server.

**TAS6011E: Requested {0} property cannot be created.**
**Explanation:**

*{0}* is the name of a property. An unexpected error occurred while creating or retrieving the named property from the database.

**User Response:**

1. Use the ParlayWebConsole to force the application server to stop.
2. Use the ParlayWebConsole to start the application server.

**TAS6012E: An error occurred. One or more of the properties files to merge was null, target is {0}, overlay is {1}.**

**Explanation:**

The component was trying to merge two Properties Objects. One or more of the Properties Objects was null.

**User Response:**

Review the logs.

**TAS6013E: An Exception occurred setting state on ManagedObject {0}.**

**Explanation:**

An Exception occurred trying to set the State on a ManagedObject.

**User Response:**

Review the Exception for possible action.

**TAS6014W: Object {0} is not a ManagedObject EJB. Ending operation.**

**Explanation:**

The operation does not proceed, because the object being acted on is not a ManagedObject.

**User Response:**

Review the ManagedObject list for inclusion of non-ManagedObjects.

**TAS6015W: No ManagedObjects were found.**

**Explanation:**

The component was not able to find any ManagedObjects in the ManagedObject list.

**User Response:**

Review the ManagedObject list, and add any ManagedObjects needing management to that list.

**TAS6017E: Properties resource {0} could not be found.**

**Explanation:**

A properties file could not be found. Defaults are used where possible.

**User Response:**

Ensure that the properties file listed is available and readable by the system.

**TAS6018E: No ApplicationManagers Objects were found.**

**Explanation:**

No JNDI entries for any ApplicationManagers were found, so no ApplicationManagers will be started.

**User Response:**

Review the JNDI entries in the properties files for the ApplicationManager to ensure they are present.

**TAS6019E: The time-out {0} value for the ApplicationManager to become ready has been exceeded.**

**Explanation:**

The allotted startup time *{0}* for the system and the ApplicationManager to become ready has been exceeded.

**User Response:**

Review the logs for errors and possible corrective action.

**TAS6020I: Current State of the {0} is {1}.**

**Explanation:**

The current State of the ManagedObject is listed.

**User Response:**

No action is needed.

**TAS6021I: The Queue status is {0}.**

**Explanation:**

The status table of the System Queue.

**User Response:**

No action is required.

**TAS6022I: The Connection table status is as follows: [{0}].**

**Explanation:**

The status table for connections.

**User Response:**

No action is required.

**TAS6023I: An Exception occured releasing the cluster wide lock. The Exception is {0}.**

**Explanation:**

An Exception occurred trying to release the cluster wide lock. This lock is used during startup and shutdown.

**User Response:**

Review the Exception for possible action.

**TAS6400I: Scheduler thread One started.**

**Explanation:**

The main Scheduler thread started successfully.

**User Response:**

None required.

**TAS6401I: Scheduler thread Two started.**

**Explanation:**

The Scheduler dispatcher thread started successfully.

**User Response:**

None required.

**TAS6402I: Scheduler thread One ended.**

**Explanation:**

The Scheduler main thread was stopped successfully.

**User Response:**

None required.

**TAS6403I: Scheduler thread Two ended.**

**Explanation:**

The Scheduler dispatcher thread was stopped successfully.

**User Response:**

None required.

**TAS6407E: Schedule event exception.**

**Explanation:**

An attempt to schedule an event resulted in an exception.

**User Response:**

- Check to make sure that the Scheduler persistent data storage location is available, accessible, and configured correctly.
- Enable trace log and reschedule event to get trace information.

**TAS6408E: Scheduler thread Two start exception.**

**Explanation:**

Exception occurred when the Scheduler tried to start the event dispatcher thread

**User Response:**

Check trace log for stack dump if trace is enabled or try to start the thread again with trace enabled.

**TAS6409E: Update scheduled event exception.**

**Explanation:**

An exception occurred when the Scheduler tried to update scheduled event with a valid key.

**User Response:**

- Check to make sure that the Scheduler persistent data storage location is available, accessible, and configured correctly.
- Enable trace log and re-execute event update to get trace information.

**TAS6410E: Delete scheduled event exception. Incorrect event key.**

**Explanation:**

An exception occurred when the Scheduler tried to delete scheduled event with a key that does not match any of the available scheduled event keys in the persistent data storage location.

**User Response:**

Supply the correct key.

**TAS6411E: Delete scheduled event exception.**

**Explanation:**

Exception occurred when the Scheduler tried to delete scheduled event with a valid key.

**User Response:**

- Check to make sure that the Scheduler persistent data storage location is available, accessible, and configured correctly.

- Enable trace log and re-execute delete event to get trace information.

**TAS6412I: Scheduler already started.**

**Explanation:**

An attempt to start the Scheduler, when it is already started.

**User Response:**

None required.

**TAS6413E: Scheduler start exception.**

**Explanation:**

An exception occurred when the Scheduler thread One start method was invoked.

**User Response:**

Check trace log for stack dump if trace is enabled or try to start the thread again with trace enabled.

**TAS6416W: Event action type {0} is not supported.**

**Explanation:**

*{1}* is the event action (create=1,delete=2,update=3) for the Scheduler. When trace is enabled, information is logged for debug.

**User Response:**

None required.

**TAS6417W: The Scheduler is currently not started.**

**Explanation:**

The Scheduler thread One is currently in a stopped state or not started. The Scheduler thread One must be running for any scheduled event being delivered.

**User Response:**

Contact the administrator responsible for starting the Scheduler thread One.

**TAS6418E: Database table access exception.**

**Explanation:**

Exception occurred when the Scheduler thread One tries to access a table in a persistent data storage location.

**User Response:**

- Check to make sure that the Scheduler persistent data storage location is available, accessible, and configured correctly.
- Enable trace log and restart thread One to get trace information.

**TAS6422E: Error parsing a collection of scheduled event entity.**

**Explanation:**

an exception occurred when the Scheduler thread One tries to parse a collection of scheduled event entity that was retrieved from a persistent data storage location.

**User Response:**

Enable the trace log to get stack information about the exception.

**TAS6430E: Object type not supported exception. Type must be a Stateless Session EJB.**

**Explanation:**

An exception occurred when the Enterprise Java Bean (EJB) handle to the application that will receive the callback notification from the Scheduler is not a supported EJB type. The Scheduler only supports Stateless session EJB for application callbacks.

**User Response:**

Use the supported EJB for applications that will be receiving notification from the Scheduler.

**TAS6431E: The EJB handle field of this ScheduledEventInfo instance is NULL.**

**Explanation:**

An exception occurred when the EJB handle to the client application that will be receiving a callback notification for the event being scheduled is not supplied. This is a required field.

**User Response:**

Supply an EJB handle of the application that will be receiving notification from the Scheduler.

**TAS6432E: The date field of this ScheduledEventInfo instance is NULL.**

**Explanation:**

An exception occurred when the time and date to deliver the event being scheduled is not supplied. This is a required field.

**User Response:**

Supply the time and date that the event will be delivered.

**TAS6433E: The key field of this ScheduledEventInfo instance is NULL.**

**Explanation:**

An exception occurred when the key associated with a Scheduled event being updated is not supplied. This field is only required for scheduled event update.

**User Response:**

Supply the associated key to the scheduled event that needs update.

**TAS6434E: ScheduledEventInfo is NULL.**

**Explanation:**

An exception occurred when the ScheduledEventInfo object, which should contain the Scheduler required information for the event being scheduled is not supplied.

**User Response:**

Supply an instance of ScheduledEventInfo with the required Scheduler information for the event being scheduled.

**TAS6435E: Repetition value is {0}. Repetition value must be equal to or greater than {1}.**

**Explanation:**

*{1}* is the number of times to repeat delivery for a scheduled event. *{2}* is the minimum required times an event should be scheduled for repetition. When trace is enabled, information is logged for debug.

**User Response:**

None required.

**TAS6436E: Interval value is {0}. Interval value must be equal to or greater than {1}.**

**Explanation:**

*{1}* is the time delay between the delivery of repeating scheduled event. *{2}* is the minimum required time delay between repeating events. When trace is enabled, information is logged for debug.

**User Response:**

Not required.

**TAS6437W: Repetition value must be greater than zero for Interval value {0} to be meaningful.**

**Explanation:**

*{1}* is the time delay between the delivery of repeating scheduled event. When trace is enabled, information is logged for debug.

**User Response:**

Not required.

**TAS6438E: ScheduledEventSettings is NULL.**

**Explanation:**

An exception occurred when the ScheduledEventSettings object, which should contain the required Scheduler information for the event being scheduled, is not supplied.

**User Response:**

Supply an instance of ScheduledEventSettings with the required Scheduler information for the event being scheduled.

**TAS6440E: Failure setting Scheduler state.**

**Explanation:**

An exception occurred when the Scheduler tried to update its state information in a persistent data storage location.

**User Response:**

- Check to make sure that the database is available, accessible, and configured correctly.
- Enable trace log and restart the Scheduler to get trace information.

**TAS6441E: Failure getting Scheduler location within a cluster.**

**Explanation:**

An exception occurred when the Scheduler tried to determine the instance of the WAS machine location that is currently active within a cluster. The location information is stored in a persistent data storage location.

**User Response:**

- Check to make sure that the database is available, accessible, and configured correctly.
- Enable trace log and restart the Scheduler to get trace information.

**TAS6442E: Failure getting Scheduler application ID.**

**Explanation:**

An exception occurred when the Scheduler tried to get an identifier for the EventScheduler EJB for trace logging. This identifier is unique for each deployed EventScheduler EJB.

**User Response:**

Enable trace log and restart the Scheduler to get trace information.

**TAS6444E: Failure sending management event ({0}) to Scheduler:{1}.**

**Explanation:**

*{1}* is a ManagedObjectEvent Type used to manage the Scheduler state. *{2}* is the WAS node location where Scheduler is currently active within a cluster. An exception occurred when the Scheduler tries to forward a management event to the active Scheduler within a cluster.

**User Response:**

- Check to make sure that the database is available, accessible, and configured correctly.
- Enable trace log and restart the Scheduler to get trace information.

**TAS6445I: The current state of {0} is {1}.**

**Explanation:**

*{1}* is the unique identifier for the Scheduler. *{2}* is the current state of the Scheduler. When trace is enabled, information is logged for debug.

**User Response:**

None required.

**TAS6446E: Error starting secondary Scheduler method.**

**Explanation:**

An exception occurred when starting a secondary Scheduler thread one within a cluster.

**User Response:**

Enable trace log and restart the Scheduler to get trace information.

**TAS6447E: Error getting class parameter type.**

**Explanation:**

An exception occurred when trying to get the object type of a given parameter within a cluster.

**User Response:**

Enable trace log and restart the Scheduler to get trace information.

**TAS6448E: Error setting Scheduler state properties.**

**Explanation:**

An exception occurred when trying to set the current state of the Scheduler within a cluster.

**User Response:**

Enable trace log and restart the Scheduler to get trace information.

# Chapter 8. Reference information for WebSphere Telecom Web Services Server

Refer to the following topics for detailed information as you use the features of Telecom Web Services Server.

## Telecom Web Services Server Standards and specifications

The IBM Telecom Web Services Server is based primarily on Parlay X Web services, which are supported for a wide range of standards specifications. Tables list the standards organizations and the various specifications that are used to build the IBM implementation.

### JCP

http://www.jcp.org

*Table 45. Java specification requests*

| Specification title | Version | Notes |
|---|---|---|
| JSR 003: Java Management Extensions (JMX) | 1.1 | Full compliance |
| JSR 053: Servlet API | 2.3 | Full compliance |
| JSR 101: Java APIs for XML based RPC | 1.0 | Full compliance |
| JSR 109: Web Services for J2EE | 1.0 | Full compliance |
| JSR 116: SIP Servlet API | 1.0 | Full compliance |
| JSR 151 :Java 2 Platform Enterprise Edition | 1.4 | Full compliance |
| JSR 153: Enterprise JavaBeans | 2.1 | Full compliance |
| JSR 168: Java Portlets | 2.1 | Full compliance |

### IETF

http://www.ietf.org

*Table 46. IETF specifications*

| Specification title | Notes |
|---|---|
| RFC 3261: SIP core protocol | Full compliance: Call Notification, Third Party Call, Terminal Status, Presence |
| RFC 3262: Reliability of provisional responses SIP | Full compliance: Third Party Call |
| RFC 3265: SIP event notification | Full compliance: Terminal Status, Presence |
| RFC 3455: Private header extensions to SIP | Full compliance: Third Party Call, Terminal Status, Presence |
| RFC 3539: Authentication, authorization, and accounting transport profile | Full compliance |
| RFC 3725: Best current practices for third-party call control (3PCC) | Partial compliance: Third Party Call supports Flow III, Flow III + PRACK, and precondition flow |

Table 46. IETF specifications (continued)

| Specification title | Notes |
|---|---|
| RFC 3863: Presence Information data format | Full compliance: Presence |
| RFC 3966: Tel URI for telephone numbers | Full compliance: Call notification, Terminal Status, Presence, Third Party Call |
| RFC 4480: Rich presence information data format | Partial Compliance: Presence |

## ETSI

http://etsi.org

Table 47. ETSI specifications

| Specification title | Version | Notes |
|---|---|---|
| ES 202 391-1: Common | 1.2.1 | Full to partial implementation: Supports all Parlay X 2.1 services (common) |
| ES 202 391-2: Third Party Call | 1.2.1 | Full implementation: Parlay X Web Services, Third-Party Call |
| ES 202 391-3: Call Notification | 1.2.1 | Full implementation to Parlay; partial implementation to SIP network (supports Notification operations, but no direction) |
| ES 202 391-4: Short Messaging | 1.2.1 | Full implementation to SMPP and partial to Parlay |
| ES 202 391-5: Multimedia Messaging | 1.2.1 | Full implementation to MM7 |
| ES 202 391-6: Payment | 1.2.1 | Partial implementation: AmountCharging and some VolumeCharging supported; no support for VolumeCharging:getAmount, ReserveAmountCharging or ReserveVolumeCharging |
| ES 202 391-7: Account Management | 1.2.1 | Partial implementation: WSDL support and Access Gateway flow |
| ES 202 391-8: Terminal Status | 1.2.1 | Full implementation: SIP and Parlay |
| ES 202 391-9: Terminal Location | 1.2.1 | Full implementation: MLP and Parlay |
| ES 202 391-10: Call Handling | 1.2.1 | Partial implementation: Supports a complete implementation to Parlay |
| ES 202 391- 11: Audio Call | 1.2.1 | Partial implementation: WSDL support and Access Gateway flow |
| ES 202 391- 12: Multimedia Conference | 1.2.1 | Partial implementation: WSDL support and Access Gateway flow |
| ES 202 391-13: Address List Management | 1.2.1 | Partial implementation: Address List Management |
| ES 202 391-14: Presence | 1.2.1 | Partial implementation: Presence Supplier support with publish operation only |
| ES 202 915-1: OSA API part 1: Overview | 1.4.1 | Partial implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |

*Table 47. ETSI specifications (continued)*

| Specification title | Version | Notes |
|---|---|---|
| ES 202 915-2: OSA API part 2: Common data definitions | 1.4.1 | Partial implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-3: OSA API part 3: Framework | 1.4.1 | Partial implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-4-1: OSA API part 4: Call Control common definitions | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-4-2: OSA API part 4: Generic Call Control SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-4-3: OSA API part 4: Multi-Party Call Control SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-4-4: OSA API part 4: Multi-Media Call Control SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-4-5: OSA API part 4: Conference Call Control SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-5: OSA API part 5: User Interaction SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-6: OSA API part 6: Mobility SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector; used by Terminal Status and Terminal Location Web services |
| ES 202 915-7: OSA API part 7: Terminal Capabilities SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-8: OSA API part 8: Data Session Control SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-9: OSA API part 9: Generic Messaging SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-10: OSA API part 10: Connectivity Manager SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-11: OSA API part 11: Account Management SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-12: OSA API part 12: Charging SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |
| ES 202 915-13: OSA API part 13: Policy Management SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector |

*Table 47. ETSI specifications (continued)*

| Specification title | Version | Notes |
|---|---|---|
| ES 202 915-15: OSA API part 15: Multi-Media Messaging SCF | 1.4.1 | Full implementation: Supports a WebSphere-based interface to vendor Parlay gateways via the Parlay Connector; used by SMS Web service |
| TS 123 140: MMS | 5.3.0 | Partial implementation: Applies to the MM7 protocol |
| TS 123 140: UMTS MMS stage 2 | 6.14.0 | Partial implementation: Supports MM7 interface for the SMS Web service |

### Open Mobile Alliance

http://openmobilealliance.org

*Table 48. OMA specifications*

| Specification title | Version | Notes |
|---|---|---|
| OMA-LIF-MLP (Mobile Location Protocol) | V3_1 | Partial implementation: Applies to a location application |
| OMA-LIF-MLP (Mobile Location Protocol) | V3_2 | Partial implementation: Applies to a location application |
| OMA-TS-XDM_Core (XML Document Management) | V1_0 | Partial implementation: As an XDMS client |
| OMA-TS-XDM_Core (XML Document Management) | V2_0 | Partial implementation: As an XDMS client |

### SMS Forum

http://www.smsforum.net

*Table 49. SMS Forum specifications*

| Specification title | Version | Notes |
|---|---|---|
| SMPP Protocol Specification | 3.4 (12-Oct-1999 Issue 1.2) | Partial implementation: Applies to SMS endpoints |

## Default Web service URIs

This topic lists default endpoint URIs used by the subcomponents of Telecom Web Services Server.

### Access Gateway

**Note:** *9080* is the default port number, but depending on your server deployment this value might be different. Refer to the **HTTP transport chain 2 port value** value in the Integrated Solutions Console.

There is one endpoint URI for each message processing flow.

Address List Management
  http://localhost:9080/AddressListManagementService/services/Group
  http://localhost:9080/AddressListManagementService/services/Member

http://localhost:9080/AddressListManagementService/services/GroupManagement

Account Management

http://localhost:9080/AccountManagementService/services/AccountManagement

Audio Call

http://localhost:9080/AudioCallService/services/AudioCall

Call Handling

http://localhost:9080/CallHandlingService/services/CallHandling

Call Notification

http://localhost:9080/CallNotificationService/services/CallDirectionManager

http://localhost:9080/CallNotificationService/services/CallDirection

http://localhost:9080/CallNotificationService/services/CallNotificationManager

http://localhost:9080/CallNotificationService/services/CallNotification

Multimedia Conferencing

http://localhost:9080/MultimediaConferenceService/services/MultimediaConference

Multimedia Messaging

http://localhost:9080/MultimediaMessageService/services/MessageNotificationManager

http://localhost:9080/MultimediaMessageService/services/MessageNotification

http://localhost:9080/MultimediaMessageService/services/ReceiveMessage

http://localhost:9080/MultimediaMessageService/services/SendMessage

Payment

http://localhost:9080/PaymentService/services/AmountCharging

http://localhost:9080/PaymentService/services/ReserveAmountCharging

http://localhost:9080/RVCPaymentService/services/ReserveVolumeCharging

http://localhost:9080/PaymentService/services/VolumeCharging

Presence

http://localhost:9080/PresenceService/services/PresenceConsumer

http://localhost:9080/PresenceService/services/PresenceNotification

http://localhost:9080/PresenceService/services/PresenceSupplier

Short Messaging Service

http://localhost:9080/ShortMessageService/services/ReceiveSms

http://localhost:9080/ShortMessageService/services/SendSms

http://localhost:9080/ShortMessageService/services/SmsNotificationManager

http://localhost:9080/ShortMessageService/services/SmsNotification

Terminal Location

http://localhost:9080/TerminalLocationService/services/TerminalLocationNotificationManager http://localhost:9080/TerminalLocationService/services/TerminalLocationNotification

http://localhost:9080/TerminalLocationService/services/TerminalLocation

ThirdParty Call

http://localhost:9080/ThirdPartyCallService/services/ThirdPartyCall

Terminal Status

http://localhost:9080/TerminalStatusService/services/TerminalStatusNotificationManager

http://localhost:9080/TerminalStatusService/services/TerminalStatusNotification

http://localhost:9080/TerminalStatusService/services/TerminalStatus

WAP Push

http://localhost:9080/WAPPUSH/services/SmsNotification

http://localhost:9080/WAPPUSH/services/SendWAPPush

http://localhost:9080/WAPPUSH/services/WapPushNotification

## Service Policy Manager

Service Policy Manager uses one Web context.

http://localhost:9080/SPM/Access/services/PolicyAccess

http://localhost:9080/SPM/Admin/services/PolicyAdministration

http://localhost:9080/SPM/Admin/services/RequesterAdministration

http://localhost:9080/SPM/Admin/services/SubscriptionAdministration

http://localhost:9080/SPM/Admin/services/ServiceAdministration

**Note:** For the Service Policy Manager user interface, use http://localhost:9080/spm/console.

## Web service implementations

Call Notification over SIP/IMS

http://localhost:9080/TWSS/ParlayX21/CallNotification/IMS/services/CallNotificationManager

http://localhost:9080/TWSS/ParlayX21/Callback/CallNotification/IMS/services/DeliveryConfirmationCallback

Presence

http://localhost:9080/TWSS/ParlayX21/Presence/IMS/services/PresenceConsumer

http://localhost:9080/TWSS/ParlayX21/Presence/IMS/services/PresenceSupplier

http://localhost:9080/TWSS/ParlayX21/Presence/IMS/services/DeliveryConfirmationCallback

Terminal Status over SIP/IMS

http://localhost:9080/TWSS/ParlayX21/TerminalStatus/IMS/services/TerminalStatus http://localhost:9080/TWSS/ParlayX21/TerminalStatus/IMS/services/TerminalStatusNotificationManager

Third Party Call over SIP/IMS

http://localhost:9080/TWSS/ParlayX21/ThirdPartyCall/IMS/services/ThirdPartyCall

Payment

http://localhost:9080/TWSS/ParlayX21/Payment/IMS/services/AmountCharging

http://localhost:9080/TWSS/ParlayX21/Payment/IMS/services/VolumeCharging

Address List Management

http://localhost:9080/TWSS/ParlayX21/AddressListManagement/IMS/services/GroupManagement

http://localhost:9080/TWSS/ParlayX21/AddressListManagement/IMS/services/Member

http://localhost:9080/TWSS/ParlayX21/AddressListManagement/IMS/services/Group

WAP Push over SMPP

http://localhost:9080/TWSS/WAP10/WAPPush/SMPP/services/SendWAPPush

http://localhost:9080/TWSS/WAP10/WAPPush/SMPP/services/WAPPushNotification

SMS over SMPP

http://localhost:9080/TWSS/ParlayX21/ShortMessaging/SMPP/services/SendSms

http://localhost:9080/TWSS/ParlayX21/ShortMessaging/SMPP/services/ReceiveSms

http://localhost:9080/TWSS/ParlayX21/ShortMessaging/SMPP/services/SmsNotificationManager

Terminal Location over MLP

http://localhost:9080/TWSS/ParlayX21/TerminalLocation/MLP/services/TerminalLocation

http://localhost:9080/TWSS/ParlayX21/TerminalLocation/MLP/services/TerminalLocationNotificationManager

MMS over MM7

http://localhost:9080/TWSS/ParlayX21/MultimediaMessaging/MM7/services/SendMessage

http://localhost:9080/TWSS/ParlayX21/MultimediaMessaging/MM7/services/MessageNotificationManager

http://localhost:9080/TWSS/ParlayX21/MultimediaMessaging/MM7/services/ReceiveMessage

http://localhost:9080/TWSS/ParlayX21/MultimediaMessaging/MM7/services/NotificationAdministrationSupport

Terminal Status over Parlay

http://localhost:9080/TWSS/ParlayX21/TerminalStatus/Parlay/services/TerminalStatus

http://localhost:9080/TWSS/ParlayX21/TerminalStatus/Parlay/services/TerminalStatusNotificationManager

Terminal Location over Parlay

http://localhost:9080/TWSS/ParlayX21/TerminalLocation/Parlay/services/TerminalLocation

http://localhost:9080/TWSS/ParlayX21/TerminalLocation/Parlay/services/TerminalLocationNotificationManager

SMS over Parlay

http://localhost:9080/TWSS/ParlayX21/ShortMessaging/Parlay/services/SendSms

http://localhost:9080/TWSS/ParlayX21/ShortMessaging/Parlay/services/ReceiveSms

http://localhost:9080/TWSS/ParlayX21/ShortMessaging/Parlay/services/SmsNotificationManager

Call Notification over Parlay

http://localhost:9080/TWSS/ParlayX21/CallNotification/Parlay/services/CallNotificationManager

http://localhost:9080/TWSS/ParlayX21/CallNotification/Parlay/services/CallDirectionManager

Third Party Call over Parlay

http://localhost:9080/TWSS/ParlayX21/ThirdPartyCall/Parlay/services/ThirdPartyCall

Call Handling over Parlay

http://localhost:9080/TWSS/ParlayX21/CallHandling/Parlay/services/CallHandling

## Service Platform components

Admission Control

http://localhost:9080/TWSS/ServicePlatform/AdmissionControl/services/AdmissionControl

Fault and Alarm

http://localhost:9080/TWSS/ServicePlatform/FaultAlarm/services/FaultAlarm

Network Resources

http://localhost:9080/TWSS/ServicePlatform/NetworkResources/services/NetworkResources

Notify Management

http://localhost:9080/TWSS/ServicePlatform/NotificationManagement/services/NotificationAdministration

http://localhost:9080/TWSS/ServicePlatform/NotificationManagement/services/NotificationRegistration http://localhost:9080/TWSS/ServicePlatform/NotificationManagement/services/NotificationStatisticsPublishing

Privacy Client

http://localhost:9080/TWSS/ServicePlatform/Privacy/services/Privacy

PX Notification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/CallNotification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/MessageNotification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/CallDirection

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/TerminalStatusNotification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/TerminalLocationNotification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/SmsNotification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/PresenceNotification

http://localhost:9080/TWSS/ServicePlatform/PXNotification/services/WAPPushNotification

Traffic Shaping

http://localhost:9080/TWSS/ServicePlatform/TrafficShaping/services/TrafficShaping

Usage Record

http://localhost:9080/TWSS/ServicePlatform/UsageRecord/services/UsageRecord

# Reference: mediation primitives

Mediation primitives consist of Java code that implement a component interface to the Access Gateway. Telecom Web Services Server supports a variety of mediation primitive types.

## Message Element Remover

Removes a data object from a service message object (SMO) using an XPath expression configuration on the mediation primitive.

### Description

The Message Element Remover mediation primitive removes a data object from a service message object (SMO) using an XPath expression corresponding to `twssHeader` (`/headers/SOAPHeader[name=twssHeaders]`). Removal of a data object can result in modification of the SOAP headers or SOAP request that is sent to the backend platform. The Message Element Remover mediation primitive also has an exception list for trusted requesters; a requester that is in the exception list will not have its data objects removed during flow processing.

Within the context of Access Gateway default flows, this mediation primitive is used to remove SOAP headers, during flow processing, that might have been mistakenly or maliciously inserted by client applications.

This mediation primitive is also used to remove sensitive SOAP headers before returning a Web service response back to the client. The exception list is used to allow trusted entities, such as the service platform when sending outbound notifications through the Access Gateway, to communicate using SOAP headers within Access Gateway flow logic. It is an optional plug-in and is used by the default Access Gateway flow:

## Policy configuration

This mediation primitive uses the following policies for runtime configuration:

None

## Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

*Table 50. Properties for the Message Element Remover mediation primitive*

| Property | Type | Promoted? | Description |
|---|---|---|---|
| request.requesterExceptionList | string | yes | A comma-delimited list of requester names whose message elements should not be removed during processing.<br>Default: /notification |
| response.requesterExceptionList | string | yes | A comma-delimited list of requester names whose message elements should not be remove during processing.<br>Default: (blank) |

Some JNDI parameters are configured for lower level objects. These parameters do not represent promoted properties.

*Table 51. JNDI parameters configured for lower level objects*

| JVM Argument | JNDI Name | Description |
|---|---|---|
| `com.ibm.websphere.sca.soap.attachments.ExpirationPolicy` | sca/attachments/ExpirationPolicy | Expiration cleanup default for attachment support specified in milliseconds.<br>Default: 30000 (5 minutes) |
| `com.ibm.ws.sca.soap.attachments.DatabaseCache` | sca/attachments/DatabaseCache | Data source name for the Database cache implementation to use for attachment support. There is no default. |

Table 51. JNDI parameters configured for lower level objects (continued)

| JVM Argument | JNDI Name | Description |
|---|---|---|
| `SOAP_ATTACHMENT_CLEANUP_THREAD_COUNT` | sca/attachments/ CleanupThreadCount | Number of scheduler threads that will be used to clean up attachments database cache table.<br>Default: 1 |
| `com.ibm.websphere.sca.soap.attachments.MaxSizeAttachmentPart` | sca/attachments/ MaxSizeAttachmentPart | Maximum size allowed for an attachment part. This is an integer number measured in MBytes (when followed by M or m), KBytes (when followed by K or k), or bytes (when followed by B or b).<br>Default: 1M |
| `com.ibm.websphere.sca.soap.attachments.MaxSizeAttachments` | sca/attachments/ MaxSizeAttachments | Maximum size allowed for all attachment parts for a given message. This is an integer number measured in MBytes (when followed by M or m), KBytes (when followed by K or k), or bytes (when followed by B or b).<br>Default: 1M |

### Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

None

### Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

None

### Message handling

Messages that are successfully processed by the Message Element Remover mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message, the error is logged and processing continues with the original SMO message.

## Transaction Identifier

Examines SOAP headers to determine if an upstream mediation primitive has supplied a global transaction ID or requester ID for a Web service request.

### Description

The Transaction Identifier mediation primitive examines SOAP headers to determine if an upstream mediation primitive has supplied a global transaction ID or requester ID for a Web service request.

If no global transaction ID exists, the Transaction Identifier mediation primitive generates a Universal Unique Identifier (UUID) (RFC 4122) and inserts it in a globalTransactionID SOAP header.

If no requester ID exists, the Transaction Identifier mediation primitive gets the WebSphere Application Server user principal for the request and inserts that ID into a requester SOAP header. Because downstream mediation primitives typically use these headers, place the Transaction Identifier mediation primitive at the start of custom Access Gateway flows.

## Policy configuration

This mediation primitive uses the following policies for runtime configuration:

None

## Mediation primitive properties

None

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

None

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:globalTransactionID>
    <!-- Global transaction ID that can be used for correlating
         transactions. If this header already exists from an
         upstream mediation primitive, this header is
         preserved and that transaction ID is used for logging.
         Otherwise, a UUID is used. -->
  </twss:globalTransactionID>
  <twss:requesterID>
    <!-- The requester ID. If this header already exists,
         it is preserved and used as the requester identity.
         Otherwise, the WAS user principal is used. If the
         requester is unauthenticated (for example, security is turned
         off), the special requester "unauthenticated" is
         used. -->
  </twss:requesterID>
  ...
</twss:twssHeaders>
```

## Message handling

Messages that are successfully processed by the Transaction Identifier mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context (″context/transient/exceptionType″) indicates whether a service-related or policy-related exception occurred. For the Transaction Identifier mediation primitive, this context is always set to `service`.

- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Contents |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code representing the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## Policy Retrieval

Fetches policy information from the Service Policy Management system and populates SOAP headers with policy information.

### Description

The Policy Retrieval mediation primitive fetches policy information from the Service Policy Management system and populates SOAP headers with policy information. This policy information is then passed along with the request to downstream mediation primitives and backend service implementations for policy-based personalization of service execution. This mediation primitive uses the access Web service interface to communicate with the Service Policy Management system.

The requester, service, and operation for the Web service request are used to retrieve policy information from the Service Policy Management system.

### Policy configuration

This mediation primitive uses the following policies for runtime configuration:

None

### Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

*Table 52. Mediation primitive properties*

| Property | Type | Promoted? | Description |
|---|---|---|---|
| servicePolicyManagerEndpoint | string | yes | The endpoint used to call the Service Policy Manager access interface.<br>Default: `http://localhost:9080/SPM/Access/services/PolicyAccess` |

*Table 52. Mediation primitive properties (continued)*

| Property | Type | Promoted? | Description |
|---|---|---|---|
| serviceIdentificationMethod | string | yes | The method used for identifying the unique service name from a Web service request. Two values/methods are supported:<br>**MESSAGE_NAMESPACE** (Default) The XML namespace used for the SOAP message contents.<br>**REQUEST_URI** Gets the full request URI from the HTTP request. This allows for differentiating between different service endpoints. |
| policyCacheInterval | long | yes | Time in milliseconds to cache of policy values for a given requester, service, and operation. A value of zero disables caching. Values must be > = 0.<br>Default: Zero by default (caching disabled) |

**Note:** If the user promotes a property, the WID tooling or the Administration Console can be used to change the property value.

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:requesterID>
    <!-- Used for the lookup of the requester's policies. If this
         header is missing, "unauthenticated" is assumed. -->
  </twss:requesterID>
  ...
</twss:twssHeaders>
```

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives. A policy header element shows for each policy retrieved from the Service Policy Manager system.

```
<twss:twssHeaders>
  ...
<serviceID>
  <!--service identification value, based on service identification method used-->
</serviceID>
  ...
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  ...
</twss:twssHeaders>
```

### Message handling

Messages that are successfully processed by the Policy Retrieval mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context ("context/transient/exceptionType") indicates whether a service-related or policy-related exception occurred.
- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Contents |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## Address Masking

Allows you to hide the real identity of the subscriber from any third-party application.

### Description

Prior to sending a message out to a third-party application, the service message object (SMO) is inspected for the presence of any address fields. The selected address fields are masked or shadowed before the contents of the SMO are sent the third-party application.

An example of the masking usage would pertain to the notifySmsReception operation of the SmsNotification interface, which is found in the short messaging (SMS) Web services.

### Policy configurations

This mediation primitive uses the following policies for runtime configuration :

**requester. MaskingEnabled**: Indicates whether masking or shadowing should be performed on the message. *True* indicates masking, shadowing, or maskingwithexpiry is enabled. And*false* indicates no masking, shadowing, or maskingwithexpiry enabled.

- Type: Boolean
- Default: False
- Allowed values: true or false

**requester.MaskingMode.Inbound**: The type of operation to be used in the request flow. It applies to all inbound flows from the third-party application.

- Type: String
- Default: Unmasking
- Allowed values:
  - masking
  - unmasking
  - shadowing
  - unshadowing
  - maskingWithExpiry
  - unmaskingWithExpiry

**requester.MaskingMode.Outbound**: The type of operation to be used in the response flow. It applies to all outbound flows to the third-party application.

- Type: String
- Allowed values:
  - masking
  - unmasking
  - shadowing
  - unshadowing
  - maskingWithExpiry
  - unmaskingWithExpiry

**address.InternalLifetime** : Length of time that the masked address is kept within the network. (Used only by the MaskingWithExpiry and UnmaskingWithExpiry operations).

- Type: Numeric
- Default: 1
- Allowed values: any positive number

**address.ExternalLifetime**: Length of time that the masked address may be used by a requester to invoke a Web service. (Used only by the MaskingWithExpiry and UnmaskingWithExpiry operations.)

- Type: Numeric
- Default: 1
- Allowed values: any positive number

**address.TimeUnit**: The time unit to be used by the address.Internal Lifetime and address.ExternalLifetime policies. For example, to configure a duration of 10 days, you would specify `address.ExternalLifetime=10` and `address.TimeUnit =days`. (Used only by the MaskingWithExpiry operation.)

- Type: TimeSpecification
- Default: Day
- Allowed values: (Millisecond)|(Second)|(Minute)|(Hour)|(Day)|(Week)|(Month)|(Year)

**address.ExternalCount**: Number of times the masked address may be used by a requester. This overrides the address.ExternalLifetime if met before time expiry is reached. (Used only by the MaskingWithExpiry and UnmaskingWithExpiry operations.)

- Type: Integer

- Default: 10
- Allowed values: any positive integer

## Masking operations

The Address Masking mediation primitive is responsible for hiding the real identifier of the subscriber (MSISDN or SIP address) from Web service client applications or third-party applications. There are situations in which you would not want to reveal a subscriber's actual address to such an application. In such cases, the address is replaced with a pseudo-address or an encrypted address value before the request is sent to the third-party application. Similarly, when a response arrives from a third-party application, your application would unmask or unshadow the address in the response to get the subscriber's actual address.

The operations to be performed are configured as policies defined to the Service Policy Manager. The field or fields to be masked/unmasked or shadowed/unshadowed are configured as XPath properties in the Address Masking mediation primitive. The mediation primitive calls the operations that are offered in the Address Masking Web service, which is a Service Platform component.

The following operations are provided:

**Masking**
Encrypts the address fields contained in a request message, using an encryption algorithm. The encrypted address value is composed of numbers, alphanumeric characters and special characters.

**Unmasking**
Decrypts one or more masked address values contained in a request message. This operation and the masking operation are complementary.

**Shadowing**
Replaces the MSISDN contained in a request message with a pseudo-MSISDN value, and returns the pseudo value to the requester. For example, the MSISDN 9818010846 might be replaced by 98180XXXXX where X is any digit from 0 to 9. The number of digits to be shadowed is user configurable.

**Note:** Pay attention to the fact that the algorithm used for shadowing is very difficult to decipher, but not impossible. Also, be aware that there is a possibility of the generated shadowed number appearing as a valid MSISDN number. Hence, Shadowing is a comparatively vulnerable option as compared to the Masking or MaskingWithExpiry operations. If you are concerned about security, you should not use the Shadowing operation.

**Unshadowing**
Replaces the pseudo-MSISDN value contained in a request message with the actual MSISDN, and returns the unshadowed MSISDN to the requester. This operation and the masking operation are complementary.

**MaskingWithExpiry**
Replaces the address fields with a pseudo (random) number, which is configured for expiry–in other words, the pseudo number can be used only for a certain period of time, after which the address expires. Any subsequent requests that use an expired number are rejected. This operation and the UnmaskingWithExpiry operation are complementary.

**UnmaskingWithExpiry**
Given a masked number, it retrieves the corresponding original number from a database and sends the response back to the user. This operation and the MaskingWithExpiry operation are complementary.

**Note:** The Address Masking component Web service mediation primitive is deployed as a Service Platform components, and the mediation primitive uses the Service Policy Manager for masking configurations. This mediation primitive is available only with the version 7.0 levels of the TWSS Access Gateway and Service Policy Manager.

**Note:** The Address Masking component Web service is deployed as a common component, and the Address Masking mediation primitive uses the Service Policy Manager for masking configurations. The version 7.0 Access Gateway and Service Policy Manager both are required prerequisites for using this feature. The Access Gateway and Service Platform components system times should be in synch at all times. This is done to prohibit any discrepancies between the expiration intervals sent from the Address Masking component Web service mediation primitive and the received Address Masking component Web service.

### Configuration parameters

Some properties must be specified in order for the mediation primitive to perform its operations. The following configuration parameters are utilized by the Address masking mediation primitive:

**XPath** This is an XPath that identifies the SMO element(s) to mask, unmask, shadow, unshadow, maskwithexpiry or unmaskwithexpiry. The XPath value should specify the exact path to the element to be masked. Multiple XPath elements are acceptable. For example; *body/getReceivedSms/senderAddress*.

**Digits** For a shadowing operation, the number of digits to replace in the MSISDN. This is a promotable property with a default value of *5*.

**Event Type**
Indicates whether the message being operated on is an inbound request message or an outbound response message. This property allows the mediation primitive to look up corresponding policies from the Service Policy Manager. Valid values are AG_INBOUND_REQUEST and AG_OUTBOUND_RESPONSE for a response flow.

**AddressMaskingEndpoint**
The endpoint URL for the Address Masking component Web service. This is a promotable property and the default value is `http://localhost:9080/AddressMaskingWeb/services/AddressMasking`.

## Transaction Recorder

Writes information about a Web service request passing through the Access Gateway to a database table for accounting purposes.

### Description

The Transaction Recorder mediation primitive writes information about a Web service request passing through the Access Gateway to a database table for accounting purposes. This data serves as a description of the transaction, including the Web service requester, transaction ID, and the service and operation invoked. The data stored in the transaction recorder table contains common transaction description information and might be referenced by other mediation primitive data tables via database foreign key relationships. The data is written to the database asynchronously, and is stored in a temporary cache that periodically updates in a batch. The data persists when this occurs.

This mediation primitive is a logical unit that has one input terminal, two output terminals, and a failure terminal. The mediation primitive can then process the message and choose where to place a response on its defined output terminals, according to the semantics of the terminal. Mediation flows are then created by wiring or connecting output terminals and input terminals.

## Policy configuration

This mediation primitive uses the following policies for runtime configuration:

**message.transaction.RecordTransaction**

Indicates whether information about this transaction should be written to the TRANSACTIONS table.

- Type: Boolean
- Default: True
- Allowed values: True or False

**message.UseUTCTime**

Indicates whether Universal Time Code (UTC), time should be used in the database record or local time.

- Type: Boolean
- Default: True
- Allowed values: True or False

## Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

**dataSourceName**

The JNDI data source name used to access the Transaction Recorder mediation primitive data table.

- Type: string
- Promoted: Yes
- Default: jdbc/TWSSDB

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:globalTransactionID>
    <!-- Global transaction ID that can be used for correlating
         transactions. If this header already exists from an
         upstream mediation primitive, this header is
         preserved and that transaction ID used for logging.
         Otherwise, a UUID is used. -->
  </twss:globalTransactionID>
  <twss:requesterID>
    <!-- The requester ID. If this header already exists,
         it is preserved and used as the requester identity.
         Otherwise, the WAS user principal is used. If the
         requester is unauthenticated (for example, security is turned
         off), the special requester "unauthenticated" is
         used. -->
  </twss:requesterID>
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  ...
</twss:twssHeaders>
```

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

None

## Data definitions

TRANSACTIONS

| Field | Key | Data type | Contents |
|---|---|---|---|
| TRANSACTIONID | PK | VARCHAR (127) | The unique transaction identifier. |
| REQUESTER | | VARCHAR (250) | The requester ID. |
| SERVICE | | VARCHAR (250) | The service being invoked. |
| SERVICEOPERATION | | VARCHAR (250) | The service operation being invoked. |
| ADDTIME | | TIMESTAMP | The transaction creation time. **See the message.UseUTCTime policy.** |

## Message handling

Messages that are successfully processed by the Transaction Recorder mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context (“context/transient/exceptionType”) indicates whether a service-related or policy-related exception occurred. For the transaction recorder mediation primitive, this context is always set to `service`.

- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Contents |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## Network Statistics

Records Web service requests, responses, entries, exiting and asynchronous write information.

### Description

The Network Statistics mediation primitive records Web service request/response entry and exit information. The statistics are stored in a database table for use by network operations. This information can be used to construct traffic summaries for network analysis and capacity planning.

### Policy configuration

This mediation primitive uses the following policies for runtime configuration:

**message.statistics.RecordStatistics**

Indicates whether statistics recording is enabled. In addition, the policy message.transaction.RecordTransaction must also be true for a record to be written.

- Type: Boolean
- Default: True
- Allowed values: True or False

**messageUTCTime**

Indicates whether Universal Time Code (UTC), time should be used in the database record or local time.

- Type: Boolean
- Default: True
- Allowed values: True or False

## Mediation primitive properties

This mediation primitive uses the following configuration properties. Some of these properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

**recordByDefault**

Indicates whether to record statistics by default in the absence of policy information.

- Type: Boolean
- Promoted: Yes
- Default: True

**eventType**

The event type to use when recording the statistics. This event type indicates where in the flow the event statistics were recorded. Each mediation primitive instance can be configured with a different event type to uniquely identify the point in the flow where the statistics were recorded.

- Type: Boolean
- Promoted: no

**dataSourceName**

The JNDI data source name used to access the Network Statistics mediation primitive table. This data source must match the data source used by the Transaction Recorder mediation primitive to satisfy foreign key relationships.

- Type: string
- Promoted: Yes
- Default: `jdbc/TWSSDB`

**messageType**

Used to indicate the type of message seen at the Network Statistics mediation primitive. Valid values are I (Inbound), O (Outbound), and F (Fault).

- Type: char
- Promoted: no

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:globalTransactionID>
    <!-- Used to identify the transaction associated with this
         request. The global transaction ID is used in a foreign
         key relationship with the TRANSACTIONS table. -->
  </twss:globalTransactionID>
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  ...
</twss:twssHeaders>
```

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

None

## Data definitions

Table name: NETWORKSTATISTICS

| Field | Key | Data type | Contents |
|---|---|---|---|
| ENTRYID | PK | VARCHAR (127) | The unique identifier for the entry. |
| TRANSACTIONID | PK | VARCHAR (127) | The unique transaction identifier. A single request/response can have multiple entries under a given transaction ID. |
| MESSAGETYPE |  | CHAR (1) | (I)nbound, (0)utbound, (F)ault. |
| EVENTTYPE |  | VARCHAR (250) | The event type indicating at what point in the flow execution this statistic is being recorded. |
| EVENTTIME |  | TIMESTAMP | The time of the message. **See the message.UseUTCTime policy.** |

Foreign key restrictions: UPDATE RESTRICT, CASCADE DELETE

View name: NETWORKSTATISTICSVIEW

| Field | Data type | Contents |
|---|---|---|
| TRANSACTIONID | VARCHAR (127) | The unique transaction identifier. |
| REQUESTER | VARCHAR (250) | The requester ID. |
| SERVICE | VARCHAR (250) | The service being invoked. |
| SERVICEOPERATION | VARCHAR (250) | The service operation being invoked. |
| MESSAGETYPE | CHAR(1) | (I)nbound, (0)utbound, (F)ault. |

| Field | Data type | Contents |
|---|---|---|
| EVENTTYPE | VARCHAR (250) | The event type indicating at what point in the flow execution this statistic is being recorded. |
| EVENTTIME | TIMESTAMP | The time of the message.**See the message.UseUTCTime policy.** |

### Message handling

Messages that are successfully processed by the Network Statistics mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context (context/transient/exceptionType) indicates whether a service-related or policy-related exception occurred. For the Network Statistics mediation primitive, this context is always set to `service`.
- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Contents |
|---|---|
| ServiceMessageObject/context/failInfo/ failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/ origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/ faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/ faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## Message Interceptor

The Access Gateway uses the Message Interceptor mediation primitive to capture message contents as they pass through, and stores them in a relational database. It also allows for the logging of specific target information contained in the message based on a pre-configured policy, and performing async write.

### Mediation primitive properties

This mediation primitive uses the following configuration property parameters. The properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console

**dataSourceName**: The JNDI data source name.

- Promoted: Yes
- Default: jdbc/MESSAGEDB

**cacheInterval**: Database batch update frequency (milliseconds).

- Promoted: Yes
- Default: 1000

**XPathMsg**: An XPath expression to the SMO which is stored in the database.

- Promoted: Yes
- Default: /body

**XPathTarget**: An XPath expression to the SMO segment which will hold the target address.

- Promoted: Yes

## Description

The Access Gateway uses the Message Interceptor mediation primitive. The Message Interceptor mediation primitive performs message capturing, as it courses through the flow. It stores requester and specific target information in addition to other attributes, when compared with the previously used WebSphere ESB Message Logger mediation primitive.

## Policy configuration

This mediation primitive uses the following policies for runtime configuration:

**message.LoggingEnabled**

Indicates whether message logging should be performed on this request.

- Type: Boolean
- Default: False

**message.Logging.Targets**

A comma separated list of target addresses to be tracked. You can add this policy in the Service Policy Manager, with the value as the targets to be tracked. By default, however, this policy is present in the Service Policy Manager.

## Considerations for migrating from TWSS version 6.2

If you used customized message processing flows in TWSS version 6.2, and if you want to preserve data that was collected by the old Message Logging mediation primitive, you will need to copy the data from the ESBLOG.MSGLOG database table WebSphere Enterprise Service Bus Message Logger mediation primitive, to the MESSAGELOG table (which is used by the new Message Interceptor mediation primitive).

Within the database tables, the comparable field names are as follows:

*Table 53.*

| Field name in version 6.2 | Field name in version 7.0 |
|---|---|
| ESBLOG.MSGLOG.TIMESTAMP | MESSAGELOG.TIMESTAMP |
| ESBLOG.MSGLOG.MESSAGEID | MESSAGELOG.GTRANID |
| ESBLOG.MSGLOG.MEDIATIONNAME | MESSAGELOG.MEDIATIONNAME |
| ESBLOG.MSGLOG.MODULENAME | MESSAGELOG.MODULENAME |

*Table 53. (continued)*

| Field name in version 6.2 | Field name in version 7.0 |
|---|---|
| ESBLOG.MSGLOG.MESSAGE | MESSAGELOG.MESSAGE |
| ESBLOG.MSGLOG.VERSION | MESSAGELOG.VERSION |
| null value | MESSAGELOG.REQUESTERID |

## Service Authorization

Provides fine-grained authorization for access to Web services.

### Description

The Service Authorization mediation primitive provides fine-grained authorization for access to Web services. Authorization is determined by examining policies indicating whether the access for the requester, service, or operation can succeed.

The Service Authorization mediation primitive follows block-or-pass semantics. All authorization policies must be `true` for the request to proceed. Multiple authorization policies are used to allow the administrator to take advantage of the policy scoping capabilities of the Service Policy Manager. Each authorization policy name implies (although it does not require) the corresponding appropriate scope for its usage.

### Policy configuration

This mediation primitive uses the following policies for runtime configuration:

| Policy | Type | Description |
|---|---|---|
| requester.service.Authorized | Boolean | Indicates whether access to a service is allowed for a requester.<br>Default: `true` |
| requester.operation.Authorized | Boolean | Indicates whether access to a service operation is allowed for a requester.<br>Default: `true` |
| requester.anonymousAccessAllowed | Boolean | Indicates whether anonymous requests (for example, unauthenticated requests) are allowed to pass.<br>Default: `false` |

### Mediation primitive properties

None

### Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <globalTransactionID>
  <!-- Used to identify the transaction associated with this request. The global transaction ID is
  </globalTransactionID>
  <requesterID>
  <!-- Used to identify the requester for this request -->
  y</requesterID>
  ...
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  ...
</twss:twssHeaders>
```

### Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

None

### Message handling

Messages that are successfully processed by the Service Authorization mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message or if the Web service request is not authorized, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context ("context/transient/exceptionType") indicates whether a service-related or policy-related exception occurred.
- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Content |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## Group Resolution

Resolves group URIs to their collection of member URIs when a Web service implementation accepts group URIs within a list of targets for a given operation.

The expanded group contents are placed into the SOAP headers for downstream processing.

This mediation primitive uses the Parlay X 2.1 Address List Management interface queryMembers() operation to perform group resolution. The Access Gateway group resolution mediator currently requires that the Address List Management group resolution service be configured. To disable group resolution, edit the SCA Module properties for the given flow and blank out the GroupResolution.endpoint. If the endpoint is not valid, the Group Resolution mediation primitive returns a fault error.

The Group Resolution mediation primitive requires the Address List Management Web service.

## Policy configuration

This mediation primitive uses the following policy for runtime configuration:

**message.groups.MaxGroupSize**

The maximum number of members that can be in a group that is returned by the group list server, for message processing to continue. If the number of members exceeds this number, the Group Resolution mediation primitive outputs a fault at its fault terminal. If this policy is not present, there is no limit to the number of group members.

- Type: integer

## Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

**groupResolutionEndpoint**

The endpoint used to call the Parlay X Address List Management interface to perform group resolution. (Refer to the topic Finding the IBM XDMS XCAP root for help with locating this value.) Default: `http://localhost:9080/TWSS/ParlayX21/AddressListManagement/IMS/services/Group`

- Type: String
- Promoted: Yes

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  ...
</twss:twssHeaders>
```

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:operationTargets>
    <!-- The number of targets for this operation after all groups
         have been resolved. Duplicates are included in this count.
         Downstream components assume a default value of '1'
         if this header is not present. -->
  </twss:operationTargets>
  <twss:resolvedGroups>
    <twss:GroupList groupURI="">
      <member>member1@domain</member>
      <member>member2@domain</member>
      ...
    </twss:GroupList>
    <twss:GroupList groupURI="">
      ...
    </twss:GroupList>
    ...
  </twss:resolvedGroups>
  ...
</twss:twssHeaders>
```

### Message handling

Messages that are successfully processed by the Group Resolution mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs in processing the message or if the maximum size of a resolved group is exceeded, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context (context/transient/exceptionType) indicates whether a service-related or policy-related exception occurred.
- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Content |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

### Compatibility

The Group Resolution mediation primitive interacts with the Parlay X Address List Manager over XCAP Web service implementation. To resolve group URIs, the Address List Manager over XCAP service interacts with the IBM WebSphere XML Document Management Server Component product (IBM XDMS).

## SLA Enforcement mediation primitives

Control the type of traffic admitted into the network for individual requesters.

## Description

The Service Level Agreement (SLA) mediation primitives–SLA Local Enforcement and SLA Cluster Enforcement–control the type of traffic admitted into the network for individual requesters.

SLA limits form a three-level hierarchy: requester, service, and operation. Requester limits govern all traffic from a specific requester across all services. Service limits constrain the rate of requests for all operations executed against that service, regardless of the distribution of requests to the individual operations.

The Telecom Web Services Server provides two types of SLA tracking in separate mediation primitives: SLA local enforcement and SLA cluster enforcement. The SLA local enforcement primitive tracks all usage information locally and assumes a distribution of requests across the Access Gateway cluster through a load-balancer, such as a round-robin scheme. The SLA cluster enforcement mediation primitive uses a distributed reservation algorithm to allocate requester rate to individual cluster members as needed, while ensuring that the requester rate limit is not exceeded across the cluster.

The Access Gateway does not make use of session affinity, as the Service Platform components does. Thus, given a random distribution of requests sprayed across an Access Gateway cluster over time, the SLA local enforcement will provide equivalent functionality to the SLA cluster enforcement mediation primitive. However, the SLA cluster enforcement mediation primitive will provide more accurate limiting of requester rate for transient, uneven distributions. This comes at the cost of additional inter-cluster traffic and computation. Both mediation primitives may also be used within the same flow, where local SLA policy limits may be set to control individual server access and cluster SLA policy limits to control access across the cluster as a whole. This method of using these primitives provides similar function to the Admission Control component Web service.

Hierarchical rate limits are expressed in policy information and may be updated in the policy database at any time. Request rates are expressed in tokens per second. A weighting can be provided for the operation as an additional policy attribute. The weighting is measured in tokens and represents the cost of executing the operation. When calculating the total cost of the operation, the SLA mediation primitives will look for an upstream SOAP header indicating the number of targets against which the operation is being executed. The total number of tokens to be consumed admitting the request is calculated using the formula: (policy weight * number of targets). This total weighting represents the number of tokens that will be consumed by allowing this request.

For SLA Cluster Enforcement, hierarchical limits in the SLA mediation primitive are enforced using a sliding window, rate limiting bucket algorithm. The algorithm works as follows: the first request received with a positive weighting from a quiesce (idle with no active sliding window) state begins a sliding window over a one second interval until the next quiesce state. Requests may arrive at any rate within the window so long as the limit is not exceeded. This constrains the average rate of Web service requests, regardless of the characteristics of the incoming flow of traffic. Requests with higher weightings will consume more tokens against the limit than requests with lower weightings. Requests with a weight of zero are automatically admitted, regardless of the limit.

For SLA Local Enforcement, hierarchical limits in the SLA mediation primitive are enforced using a fixed window algorithm. When the system receives a request, it calculates the SLA usage by computing the total tokens used in that particular time window. If the total tokens consumed exceeds the limit, the request is rejected.

## Policy configuration

This mediation primitive uses the following policies for runtime configuration.

For the SLA Local Enforcement mediation primitive:

**message.sla.LocalEnabled**

Indicates whether to perform SLA management on this request.
- Type: Boolean
- Default: True

**message.sla.LocalWeight**

The cluster enforcement weighting for this message. Values should be greater than or equal to zero.
- Type: integer
- Default: 0

**message.sla.LocalRequesterRate**

The rate of traffic for this requester rate in tokens per second.
- Type: integer
- Default: 0 (denied)

**message.sla.LocalServiceRate**

The rate of traffic for this service in tokens per second.
- Type: integer
- Default: 0 (denied)

**message.sla.LocalOperationRate**

The rate of traffic for this operation in tokens per second.
- Type: integer
- Default: 0 (denied)

For the SLA Cluster Enforcement mediation primitive:

**message.sla.ClusterEnabled**

Indicates whether to perform SLA management on this request.
- Type: Boolean
- Default: True

**message.sla.ClusterWeight**

The cluster enforcement weighting for this message. Values should be greater than or equal to zero.
- Type: integer

- Default: 0

**message.sla.ClusterRequesterRate**

The rate of traffic for this requester rate in tokens per second.
- Type: integer
- Default: 0 (denied)

**message.sla.ClusterServiceRate**

The rate of traffic for this service in tokens per second.
- Type: integer
- Default: 0 (denied)

**message.sla.ClusterOperationRate**

The rate of traffic for this operation in tokens per second.
- Type: integer
- Default: 0 (denied)

For both SLA Local Enforcement and SLA Cluster Enforcement mediation primitives:

**message.sla.TimeUnit**

Measures the time window. This policy has to be used in combination with "message.sla.TimeWindow".
- Type: String
- Default: Second supported values are "Day", "Hour", "Minute", and "Second"

**message.sla.TimeWindow**

The amount time for which the Requester/Service/Operation rate is valid. This policy has to be used in combination with "message.sla.TimeUnit".
- Type: Long
- Default: 600

## Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console

**cleanupInterval**

The interval between cleanup timer runs. The cleanup timer cleans up inactive requesters from the in-memory data structures, which is used to track SLA usage. The interval is measured in milliseconds.
- Type: long
- Default: 2000
- Promoted: Yes

**maxEntriesPerCleanup**

The maximum number of entries to examine per run of the cleanup timer. As there can be a large number of requesters tracked within the in-memory data structures, this spreads out the cost of discovering and cleaning up inactive requesters over time. The higher this number, the faster inactive requesters are purged from the data structures (more efficient with memory), but at the cost of higher CPU utilization.

- Type: integer
- Promoted: Yes
- Default: 25

The SLA Cluster Enforcement mediation primitive includes the following additional configuration properties:

**requestTimeoutInterval**

The timeout interval for requests to the active coordinator. This interval is measured in milliseconds.

- Type: long
- Promoted: Yes
- Default: 2000

**reservationExpiration**

The time after which a reservation is considered to expire provided that the active coordinator has not received any messages from a cluster member. A reservation only expires due to this interval if a member of the cluster has failed. This time is measured in milliseconds.

- Type: long
- Promoted: Yes
- Default: 1000

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

```
<twss:twssHeaders>
  ...
  <twss:requesterID>
    <!-- The requester ID. If this header is missing, then
         "UNAUTHENTICATED" is assumed. -->
  </twss:requesterID>
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  <twss:operationTargets>
    <!-- Used to calculate the weighting or cost of this request. If
         this header is not provided, then a default of '1' is
         assumed. -->
  </twss:operationTargets>
  ...
</twss:twssHeaders>
```

## Added SOAP headers

None

## Message handling

Messages that are successfully processed by the SLA mediation primitives are passed to the output terminal of the mediation primitive. If an error occurs while processing a message or an SLA has been exceeded, the message is redirected to the fault terminal:

- The service message object (SMO) data object transient context ("context/transient/exceptionType") indicates whether a service-related or policy-related exception occurred.
- Fault information is set in the SMO headers as indicated in the following table:

| SMO header (represented by XPath) | Content |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## Reservation scheme

This section applies only to the SLA Cluster Enforcement mediation primitive.

A reservation scheme is used to allocate rate amongst members of a cluster to enforce cluster-wide limits. A single coordinator is elected among the cluster members. The coordinator maintains reservation information for each cluster member against the cluster-wide limits. Cluster members send reservation requests as needed (for example, when insufficient rate is currently allocated) to attempt to admit incoming requests. When the allocated rate is no longer needed, a release request is sent to the coordinator relinquishing rate. To determine when to release rate, a rate estimator is associated with each operation. A running average of the current traffic is maintained and used to determine when rate is no longer needed. Rate is released gradually (every second) in chunks that are a percentage of the cluster-wide rate limit.

To scale up to many different requesters, the SLA Cluster Enforcement mediation primitive only performs reservation against the requester rate limit across the cluster. Service and operation rate limits are tracked locally. This reduces the number of estimators and intercluster messaging required. In addition, as policy information containing the rate limit information is provided with each message, the SLA Cluster Enforcement mediation primitive implements a memory management scheme whereby inactive requester limits are forgotten and limits rediscovered upon the receipt of additional traffic.

Reservation requests for additional requester rate might be denied at any time due to lack of remaining rate. When a node member receives a rejected reservation request, it enters a silence period. This silence period suppresses additional reservation requests, reducing intercluster traffic and allowing the network to settle before attempting additional reservation. In steady state, this minimizes the intercluster messaging required.

Upon failover of the node housing the cluster coordinator, a new coordinator is elected. A message is sent to all nodes resetting their reservation status. This results in a ″reset″ of the algorithm without interrupting service. Inter-cluster traffic might increase temporarily while the algorithm rebuilds state to reduce intercluster messaging.

### Guidelines for choosing limits

There are no guidelines for choosing limits.

When selecting cluster-wide limits for the SLA Cluster Enforcement mediation, primitive reservation requests are only made on an as-needed basis.

## Service Invocation

Performs runtime selection of which backend service implementation to invoke.

### Description

The Service Invocation mediation primitive performs runtime selection of which backend service implementation to invoke. The backend service endpoint is determined by a policy attribute. The Service Invocation mediation primitive also provides additional function around setting up the call to the backend Web service, including session and attachment propagation.

### Policy configuration

This mediation primitive makes use of the following policies for runtime configuration:

**service.Endpoint**

The endpoint used for calling the backend Web service. This attribute identifies the service implementation mapping for the Web service request. If this policy is critical for calling the backend service, implementation and fault are raised when this policy is not present.

- Type: String

**service.notification.Username**

User name used for transport-level security in delivering outbound notifications. If this policy is not present, no transport-level security is used.

- Type: String

**service.notification.Password**

Contains an encrypted password used for transport-level security in delivering outbound notifications. If this policy is not present, no transport-level security is used. Encrypt the password with the AdminTool that is provided with TWSS: java -jar AdminTool.jarencryptpassword.

- Type: String

## Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

**propagateAttachments**

Indicates whether to propagate attachments along with the backend Web service invocation. If false, no attachments are included with the request. This allows this primitive to be reused in different parts of the flow to provide dynamic invocation capabilities where it might not be desirable to propagate attachments. This capability is enabled by default.

- Type: Boolean
- Default: True
- Promoted: Yes

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives. The notification element is expected only for notification messages:

```
<twss:twssHeaders>
  ...
  <twss:policies>
    <twss:policy attribute="" value=""/>
    <twss:policy attribute="" value=""/>
    ...
  </twss:policies>
  <twss:notification>
    <twss:destination>
      <!-- The notification endpoint destination -->
    </twss:destination>
  </twss:notification>
  ...
</twss:twssHeaders>
```

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

None

## Message handling

Messages that are successfully processed by the Service Invocation mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs in processing the message, the message is redirected to the fault terminal.

- The service message object (SMO) data object transient context (“context/transient/exceptionType”) indicates whether a service-related or policy-related exception occurred,

- Fault information is set in the SMO headers as indicated in the following table:

| SMO Header (Represented by XPath) | Contents |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originated the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |

## JMX Notification

Used to emit Java Management Extensions (JMX) notifications by means of the WebSphere Application JMX infrastructure.

### Description

The JMX Notification mediation primitive is used to emit Java Management Extensions (JMX) notifications via the WebSphere Application JMX infrastructure. WebSphere Application Server provides plug-points that IBM and third-party software use to receive those notifications and gather management-related information. The JMX Notification mediation primitive gathers information about Access Gateway processing faults from the service message object (SMO) headers according to the SMO schema.

### SMO headers

The headers used by other Access Gateway mediation primitives are described in the property table:

| SMO header (represented by XPath) | Content |
|---|---|
| ServiceMessageObject/context/failInfo/failureString | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/failInfo/origin | The name of the mediation primitive class that originates the fault. |
| ServiceMessageObject/SOAPFaultInfo/faultcode | The TWSS message code that represents the fault situation. For example, `SOAC4025E`. |
| ServiceMessageObject/SOAPFaultInfo/faultstring | The full message text that represents the fault situation with substituted variables. For example, `SOAC4025E: Error occurred`. |
| ServiceMessageObject/context/transient/exceptionType | The exception type element under the transient context may contain one of two values: `service` or `policy`. These values represent the context of the fault: indicating whether it was caused by the operation of the service or a policy-related action. |

The notification contains the detailed fault string with the origin element used as the originating class for the JMX notification.

When running in a secured environment, the JMX Notification mediation primitive must have access to the administrative user name and password to have sufficient privileges to emit notification events. These credentials can be configured as part of the mediation primitive configuration properties.

## Policy configuration

This mediation primitive uses the following policies for runtime configuration:

None

## Mediation primitive properties

This mediation primitive uses the following configuration properties. These properties can be modified using WebSphere Integration Developer (WID) tooling. Properties that are promoted can be configured using the Integrated Solutions Console.

| Property | Type | Promoted? | Description |
|---|---|---|---|
| adminUserName | string | yes | The administrative user name used to run as the administrative user principal when emitting JMX notifications. This user name is needed only when security is enabled.<br>Default: (blank) |
| adminPassword | string | yes | The administrative user password used to run as the administrative principal when emitting JMX notifications. Encrypt the password with the AdminTool provided with TWSS: **java -jar AdminTool.jar encrypt***password*<br>Default: (blank) |

## Upstream SOAP headers

The following SOAP header elements are expected from upstream mediation primitives:

None

## Added SOAP headers

The following SOAP header elements are added or modified for downstream mediation primitives:

None

## Message handling

Messages that are successfully processed by the JMX Notification mediation primitive are passed to the output terminal of the mediation primitive. If an error occurs while processing the message, the error is logged for tracing.

## CEI Event Emitter

The Access Gateway uses the Event Emitter mediation primitive provided with the base WebSphere ESB platform for emitting Common Event Infrastructure (CEI) events.

### Description

The Access Gateway uses the Event Emitter mediation primitive provided with the base WebSphere ESB platform for emitting Common Event Infrastructure (CEI) events.

To allow for enabling or disabling Event Emitter output through policies within the default flow, an additional filtering step has been added in front of the Event Emitter mediation primitive. This additional filtering step determines whether to emit the event. The policy configuration described in this topic applies only to the Access Gateway default flow.

For more information, refer to the topic *Event Emitter mediation primitive* in the WebSphere ESB information center.

### Policy configuration

This mediation primitive uses the following policy for runtime configuration:

**message.cei.Enabled**

Indicates whether the CEI Event emission is enabled.
- Type: Boolean
- Default: True

### Troubleshooting

After you start the cluster member in WebSphere ESB, the following warning message may be issued. A full description for the message can be found in the WebSphere Application Server information center.

**ADMU3100I**: Reading configuration for server: server1 Warning: Invalid ws.ext.dir value *${CEI_HOME}/lib:${CEI_HOME}/client* found and will not be used to start this server. Correct this value and restart the server. Could not expand variable specified.

# Reference: default policies for the Access Gateway and Web service implementations

TWSS includes default policies for Access Gateway and for many of the Web service implementations sets.

You initialize the default policies by running scripts after you install the Service Policy Manager.

For more information, see the topic *Initializing policies*.

## Access Gateway

The Access Gateway makes use of mediation primitives, many of which have default policies defined for them.

The following mediation primitives have default policies:

- Transaction Recorder
- Service Invocation
- Network Statistics
- Message Interceptor
- Service Authorization
- SLA Enforcement

  **Note:** The SLA Local Enforcement mediation primitive will be deprecated after 7.0.
- Group Resolution
- CEI Event Emitter
- Address Masking

For a list of the default policies defined for each one, see *Mediation primitives* in the Reference section of this information center.

## Default policies for Web service implementations based on SIP/IMS

Telecom Web Services Server supports Web service implementations that operate over SIP/IMS.

### Default service policies for Parlay X Call Notification over SIP/IMS

There are no default service polices for Parlay X Call Notification over SIP/IMS.

### Default service policies for Parlay X Presence over SIP/IMS

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Presence over SIP/IMS in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Presence over SIP/IMS parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by prefixing it with `service.config` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

#### service.config.presence.PresenceServerSIPURI

If the `PresenceServerSIPURI` setting is not empty, the outbound SUBSCRIBE request will have a topmost Route header containing the PresenceServerSIPURI. The SIP Container will direct the request according to this Route header.

- Type: String
- Value: null

### service.standard.MaximumNotificationDuration

The maximum amount of time a notification may be set up for. Smaller values are more restrictive.

- Type: Long
- Value: 2^63-1

### service.config.presence.PresenceServerResourceName

The resource name to use when communicating with the Traffic Shaping component.

- Type: String
- Value: `PresenceServer`

### service.config.ServiceImplementationName

Used for Fault and Alarm component Web service and Usage Record component Web service. Must be 127 characters or less in length.

- Type: String
- Value: `PX21_PRS_IMS`

### service.config.presence.PresenceFetchTimeoutInMillis

Used for a SIP-based fetch or CheckImmediate operation. Fetches taking longer than this duration will receive a NotRetrieved RetrievalStatus. The value is specified in milliseconds and must be greater than zero.

- Type: integer
- Default Value: `1000`

### service.config.presence.SubscribePresenceTimeout

Maximum lifetime of state due to a subscribePresence request. Smaller values are more restrictive.

- Type: integer
- Value: 2^63 - 1

### service.standard.MaximumNotificationFrequency

Maximum rate of notification delivery in milliseconds (also can be considered minimum time between notifications). Smaller values are more restrictive.

- Type: integer
- Value: `1`

### service.config.DefaultNotificationDuration

Default duration for a startPresenceNotification operation. Specified in milliseconds. Smaller values are more restrictive.

- Type: integer
- Value: `86400000` (24 hours)

### service.standard.MaximumCount

Maximum number of notifications that may be requested. Smaller values are more restrictive.

- Type: integer
- Value: `2^31 - 1`

### service.standard.UnlimitedCountAllowed

Allowed to specify unlimited notification count (either by not specifying the optional Count message part in StartNotificationRequest or by specifying a value of zero). False values are more restrictive.

- Type: Boolean
- Value: `true`

### service.standard.GroupSupport

Groups may be included with addresses.

- Type: Boolean
- Value: `true`

### service.standard.NestedGroupSupport

Defines if nested groups are supported in group definitions.

- Type: Boolean
- Value: `true`

### service.config.ims.InterOperatorIdentifier

The setting may be either an empty string or a gen-value per RFC 3261 Section 25. If not empty, the service implementation includes P-Charging-Vector headers in its SIP signaling.

- Type: String
- Value: null

### service.config.AdditionalGroupURISchemes

While presentities may only have SIP, SIPS, or TEL URI schemes, group definitions may use other URI schemes. This setting specifies a comma-separated list of URI schemes that the service treats as indicating a group URI. This setting must be coordinated with the Group List Manager component configuration.

- Type: String
- Value: null

### service.config.presence.supplier.PublishPresenceTimeout

The expiration time for Presence information documents, which are stored in the Presence server.

- Type: integer
- Value: 60 (Seconds)

### service.config.presence.supplier.SubscribePresenceWinfoTimeout

The expiration timeframe for the SIP SUBSCRIBE requests watcher information.

- Type: integer
- Value: 60 (Seconds)

### service.config.presence.supplier.MaximumPublishDurationAllowed

If PUBLISH request is not processed within the specified time, the application responds with the Request Timeout to the Web Service request.

- Type: integer
- Value: 60 (Seconds)

## Default service policies for Parlay X Terminal Status over SIP/IMS

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Terminal Status over SIP/IMS in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Terminal Status over SIP/IMS parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by prefixing it with `service.config` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

### service.config. ProcessUpToLimit

If set to `true`, the MaximumTarget limits specified is enforced.

- Type: Boolean
- Value: `true`

### service.config. MaximumTargets

Number of targets processed for the ProcessUpToLimit policy, if the GetStatusForGroup operation is set to true.

- Type: Integer
- Value: `5`

### service.config.presence.PresenceServerSIPURI

If the `PresenceServerSIPURI` setting is not empty, the outbound `SUBSCRIBE` request will have a topmost Route header containing the `PresenceServerSIPURI`. The SIP Container will direct the request according to this Route header.

- Type: string
- Value: null

### service.config.presence.PresenceServerResourceName

The resource name to use when communicating with the Traffic Shaping component.

- Type: string
- Value: `PresenceServer`

### service.config.ServiceImplementationName

Used for Fault and Alarm component Web service and Usage Record component Web service. Must be 127 characters or less in length.

- Type: string
- Value: `TerminalStatus_IMS =  PX21_TS_IMS`

### service.config.presence.TerminalBusyRpidActivities

A comma-separated list of the RPID activities that indicate a terminal is busy.

- Type: string
- Value: `on-the-phone`

### service.config.presence.PresenceFetchTimeoutInMillis

For a SIP-based fetch or CheckImmediate operation. Fetches taking longer than this duration will receive a NotRetrieved RetrievalStatus. The value is specified in milliseconds and must be greater than zero.

- Type: integer
- Value: `1000`

### service.standard.BusyAvailable

Defines whether `busy` can be returned as a status or be a trigger.

- Type: Boolean
- Value: `true`

### service.standard.MaximumNotificationFrequency

Maximum rate of notification delivery in milliseconds (also can be considered minimum time between notifications). Smaller values are more restrictive.

- Type: integer
- Value: `1`

### service.standard.MaximumNotificationDuration

Maximum amount of time a notification can be configured for. Smaller values are more restrictive.

- Type: integer
- Value: `3600000` (1 hour)

### service.standard.MaximumNotificationAddresses

Maximum number of addresses for which a notification can be set up. Smaller values are more restrictive.

- Type: integer
- Value: `50`

### service.config.DefaultNotificationDuration

Default duration for a startPresenceNotification operation. Specified in milliseconds. Smaller values are more restrictive.

- Type: integer
- Value: `86400000` (24 hours)

### service.standard.MaximumCount

Maximum number of notifications that can be requested. Smaller values are more restrictive.

- Type: integer
- Value: `10`

### service.standard.UnlimitedCountAllowed

Allowed to specify unlimited notification count (either by not specifying the optional Count message part in StartNotificationRequest or by specifying a value of zero). False values are more restrictive.

- Type: Boolean
- Value: `true`

### service.standard.GroupSupport

Defines whether groups can be included with addresses.

- Type: Boolean
- Value: `true`

### service.standard.NestedGroupSupport

Defines whether nested groups can be included in group definitions.

- Type: Boolean
- Value: `true`

### service.config.ims.InterOperatorIdentifier

The setting can be either an empty string or a generated value, per RFC 3261 Section 25. If it is not empty, the service implementation includes P-Charging-Vector headers in its SIP signaling.

- Type: string
- Value: null

### service.config.AdditionalGroupURISchemes

While presentities may only have SIP, SIPS, or TEL URI schemes, group definitions may use other URI schemes. This setting specifies a comma-separated list of URI schemes that the Web service treats as indicating a group URI. This setting must be coordinated with the Group List Manager component configuration.

- Type: string
- Value: null

## Default service policies for Parlay X Call Handling over SIP/IMS

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X Call Handling over SIP/IMS in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The following service policies can be established for Parlay X Call Handling over SIP/IMS Mbean configuration attributes.

### service.config.callhandling.AudioContentAvailable

Accepts audio content for playing with the audio player.
- Type: Boolean
- Value: `True`

### service.config.callhandling.AudioFormatsSupported

A comma-separated string of audio formats (for example WAV, MP3, AU).
- Type: String
- Value: `WAV`

### service.config.callhandling.TextToSpeechAvailable

Accepts text as an input for processing text-to-speech.
- Type: Boolean
- Value: `True`

### service.config.callhandling.VoiceMailAvailable

Indicates whether voice mail is available.
- Type: Boolean
- Value: `True`

### service.config.callhandling.VoiceXmlAvailable

Accepts VoiceXML for processing with a VoiceXML browser.
- Type: Boolean
- Value: `True`

### service.config.standardGroupSupport

Indicates whether groups may be included with addresses.
- Type: Boolean
- Value: `False`

### service.config.NestedGroupSupport

Indicates whether nested groups are supported in group definitions.
- Type: Boolean
- Value: `False`

## Default service policies for Parlay X Third Party Call over SIP/IMS

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Third Party Call over SIP/IMS in the SOAP headers. If the policies are not found in the Telecom Web Services Server SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Third Party Call over SIP/IMS parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by prefixing it with `service.config.ThirdPartyCall_IMS.` and including the resulting header within the incoming SOAP header. Policy-based settings override MBean settings.

### service.config.ims.InterOperatorIdentifier

Used for Fault and Alarm component Web service and Usage Record component Web service. To provision the Inter Operator Identifier (IOI), which is necessary for using the charge header support vector utility, set this value equal to the domain name for the S-CSCF.

- Type: string
- Value: `ibm.com`
- Default: null

### service.config.sip.MediaServerURI

The SIP URI for the SIP endpoint that will serve as a media server (for example, play audio to the phone while the call is being connected).

- Type: string
- Value: `sip:myMediaServer@ibm.com`
- Default: null

### service.config.ims.PHeaderEnabled

This indicates whether to insert a p-asserted-identity and p-charging-vector in outbound SIP messages. Used in IMS.

- Type: Boolean
- Value: `true/false`
- Default: `false`

### service.config.sip.GeneratedTrafficEstimate

Estimate for message traffic for Parlay X Third Party Call over SIP/IMS for the method being called; for example, the ratio is 1:50 for `makecall`.

- Type: int
- Value: `50`
- Default: `50`

### service.config.sip.OutboundProxy

URI of the outbound proxy for Parlay X Third Party Call over SIP/IMS.

- Type: string
- Value: `sip:scscf.somedomain.com:5060; orig; transport=udp`
- Default: null

### service.config.sip.OutboundProxyResourceName

Resource name of the outbound proxy for Parlay X Third Party Call over SIP/IMS.

- Type: string
- Value: `OutgoingProxyResSpec`

- Default: `OutgoingProxyResSpec`

### service.config.StatusRetainTime

The number of minutes to keep status records in the database.

- Type: long
- Value: `2000`
- Default: `43200` (30 days)

# Default policies for Web service implementations based on IBM WebSphere software

Telecom Web Services Server (TWSS) supports Web service implementations that interoperate with other components within the IBM WebSphere product group–including WebSphere Application Server, IBM XDMS, and other instances of TWSS.

## Default service policies for Parlay X Payment (PostPaid)

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Payment (PostPaid) in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Payment (PostPaid) parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by prefixing it with `service.config` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

### service.config.PaymentUsageRecordEnabled

Determines whether payment records will be written to the service usage database.

- Type: Boolean
- Value: `true`

### service.config.ResultsUsageRecordEnable

Determines whether results records will be written to the service usage database.

- Type: Boolean
- Value: `false`

### service.config.CEIEnabled

Determines whether payment records will be written to the CEI database.

- Type: Boolean
- Value: `false`

### service.config.Currency

Determines whether the billing mediator will resolve the currency.

- Type: string
- Value: null

#### notification.Endpoint.Type

Determines the type of import component to which your application connects. If the value is set to OneAPI, the message is routed to the callout corresponding to the Payment_OneAPIImport. If the value is set to JSON, the message is routed to the callout corresponding to Payment_JSONImport. If the value is null, then the application uses the import component that is defined for the Web service your application is associating with.

- Type: string
- Value: null

### Default service policies for Parlay X Address List Manager over XCAP

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X Address List Manager over XCAP in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The following policies are configured through the Service Policy Manager. These policies are expected to be in the SOAP header. They may be stored elsewhere, however.

#### service.config.UserName

Use this policy to supply a user name when security is enabled on the XCAP server but not on the local server. The user name is passed to the XCAP server for basic authentication or to provide an asserted identity, as needed.

This policy is normally used only in a development or test environment. It is not used when security is enabled on the server where Parlay X Address List Manager over XCAP is running.

- Type: string
- Value: (null)
- Default: none

#### service.config.auth.Password

If the XCAP server requires a password, you must set a password for each requester in the Service Policy Manager. This password must be encrypted using the admin tool before it is entered in the Service Policy Manager.

- Type: string
- Value: (null)
- Default: none

## Default policies for Web service implementations based on Direct Connect protocols

Telecom Web Services Server supports Web service implementations that operate over Direct Connect protocols such as SMPP, MLP, and MM7.

### Default service policies for WAP Push over SMPP

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to WAP Push over SMPP in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The following polices are configured through the Service Policy Manager. These policies are expected to be in the SOAP header. They may be stored elsewhere, however.

### service.standard.ChargingSupported

If true, then charging is supported for send operations. This flag is for turning on and off the data being captured in usage records.
- Type: Boolean
- Value: `true`

### service.config.MaximumTargets

The maximum number of targets to permit for one request This number is a positive number, ranging in value from 0 (no maximum) to the maximum value for the data type.
- Type: integer
- Value: `5`

### service.config.ProcessUpToLimit

Specifies how to proceed when the maximum number of targets has been exceeded. If true, then the message is sent to the first number of target addresses up to the maximum, with the delivery status for the remaining target addresses being marked as undeliverable. If false, then the entire request is refused and a PolicyException is generated.
- Type: Boolean
- Value: `true`

### service.config.CallLevelPrivacy

Specifies how to proceed when one target address fails the privacy check. If true, then if any target address fails the privacy check, the entire request is refused, and the delivery status for every target addresses is marked as undeliverable. If false, then the message is sent to the target addresses that pass the privacy check.
- Type: Boolean
- Value: `false`

### service.config.RequestPriority

The priority of the message on the JMS queue (0-4 normal, 5-9 expedited).
- Type: string
- Value: `0`

### service.config.messaging.ConfirmDelivery

If true, then delivery confirmation is requested before making a final update to the message status (Delivered or DeliveryImpossible).
- Type: Boolean
- Value: `false`

### service.config.StatusRetainTime

Minimum time, in minutes, to retain the results of the message send.

- Type: integer
- Value: `1440` (24 hours)

### service.config.wap.MessageValidTime

Maximum time, in minutes, that the WAP Push message will remain valid after being received in the user handset.

- Type: integer
- Value: `43200` (30 days)

### service.config.wap.PushAction

The action to be taken when the SI-WAP Push is received by the handset.

- Type: string
- Value: `Signal-medium`

### service.config.wap.UsePushActionFromUser

Specifies where the Web service implementation obtains the action value. If true, the Web service implementation code uses the action value provided by the user through the WAP Push WSDL. If false, the action value is obtained from the Service Policy Manager.

- Type: Boolean
- Value: `false`

### service.config.messaging.Billing

The charging code to be passed to the backend (SMSC) along with the request for billing purposes. This policy is used only when the WAP Push over SMPP and Parlay X SMS over SMPP Web service implementations are coresident on the same server. Not required.

- Type: string
- Value: null

### service.custom.messaging.ServiceType

The type of messaging service to use, for billing purposes. Not required.

- Type: string
- Value: null

## Default service policies for Parlay X SMS over SMPP

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X SMS over SMPP in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X SMS over SMPP parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by using the prefix `service.config.SMS_IMS.` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

### service.standard.VirtualNumber

The number that will be used as source address in Parlay sendSms call. In Parlay, this need to be valid address in the network; otherwise the sendSms() call will fail. Not required.

- Type: string
- Value: `tel:111112223300`

### service.standard.MaximumNotificationAddresses

Maximum number of addresses for which a notification can be set up. For some services, this limit is used only if `service.config.ProcessUpToLimit` is true. Not required.

- Type: integer
- Value: `5`

### service.config.ChargeForFirstSegmentOnly

If true, then charging information for large messages is included only in the first segment. If false, then charging information is included in all segments. (In the case of a single-segment message, charging information is always included in the segment.) Required.

- Type: Boolean
- Value: `true`

### service.config.messaging.ConfirmDelivery

Causes the final status to be delayed for a send request until the backend reports back on each target. The ReceiptRequest is a request for a notification on the results of the send request from Parlay X SMS over SMPP back to the requester, and it requires that the policy be true in order for the notification to have meaning and for the notification to be sent. Not required.

- Type: Boolean
- Value: `false`

### service.config.MaximumTargets

The maximum number of targets the Web service implementation should use in a single message. This number is used by the common layer if `service.config.ProcessUpToLimit` is set to true. Not required.

- Type: integer
- Value: `5`

### service.config.ProcessUpToLimit

If true, then enforce `service.config.MaximumTargets`. Not required.

- Type: Boolean
- Value: `true`

### service.config.ReceivedRetainTime

Time in minutes to keep the received data messages in the database. Not required.

- Type: integer
- Value: `43200` (30 days)

### service.config.StatusRetainTime

Time in minutes to keep the send data status records in the database. Not required.

- Type: integer
- Value: `43200` (30 days)

### service.config.messaging.target.Aliases

The list of valid alias names currently configured, with target ranges separated by semicolons. Not required.

- Type: SemiColonStringList
- Value: null

### service.config.messaging.target.Ranges

The list of target URI ranges separated by semicolons. Not required.

- Type: SemiColonStringList
- Value: null

### service.config.messaging.Billing

The charging code to be passed to the backend (SMSC) along with the request for billing purposes. This policy is used only when the WAP Push over SMPP and Parlay X SMS over SMPP Web service implementations are coresident on the same server. Not required.

- Type: string
- Value: null

### service.custom.messaging.ServiceType

The type of messaging service to use, for billing purposes. Not required.

- Type: string
- Value: null

### service.custom.priorityEnabled

Whether priority is enabled. Use in conjunction with service.custom.requestPriority to define the priority of messages according to which backend servers the messages are associated with.

- Type: Boolean
- Value: `true`

### service.custom.requestPriority

The priority value that is set. When service.custom.priorityEnabled is true, this policy defines the priority of messages for SMS over SMPP. Different accounts with different priorities are created in the network element (SMSC). For different accounts, only the user IDs and passwords of the defined backend servers vary. IP address, port number, and other details remain the same.

- Type: integer
- Value: 1

To define priority levels, use `service.custom.priority` with the following syntax:

```
service.custom.priority.backend_alias.priority.Primary = user,password
service.custom.priority.backend_alias.priority.Secondary = user,password
```

where:

*backend_alias* is the alias for the backend.

*priority* is the priority level; it matches the value for `service.custom.requestPriority`.

*user* is the user ID for the backend.

*password* is password for the backend user ID.

For example, to set priority levels for two backend aliases:

```
service.custom.priority.backend1.1.Primary = user,password
service.custom.priority.backend1.1.Secondary = user,password
service.custom.priority.backend2.1.Primary = user,password
service.custom.priority.backend2.1.Secondary = user,password
```

### notification.Endpoint.Type

Determines the type of import component to which your application connects. If the value is set to OneAPI, the message is routed to the callout corresponding to the SMS_OneAPIImport. If the value is set to JSON, the message is routed to the callout corresponding to SMS_JSONImport. If the value is null, then the application uses the import component that is defined for the Web service your application is associating with.

- Type: string
- Value: null

## Default service policies for Parlay X Terminal Location over MLP

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Terminal Location over MLP in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Terminal Location over MLP parameters found in the TWSS Administration Console, can be supplied on a per-requester or per-operation level by prefixing them with the service.config.TerminalLocation_MLP. This includes both the resulting and SOAP headers. The policy-based setting overrides, will allow only those policies which are more restrictive than console settings.

### service.Endpoint

The target endpoint URI of the backend service.

- Type: String

### service.config.location.MaximumAge

The maximum acceptable age in seconds, of the returned location information.

- Type: Integer (0)
- Value: `0 to MAX_INT`

### service.config.ResponseTime

The maximum time that it takes the application to wait for a response.

- Type: Integer (0)
- Value: `0 to MAX_INT`

### service.config.DelayTolerance

Response time versus accuracy. Must be one of the following: `NoDelay` `LowDelay` `DelayToTolerant`.

- String: `DelayToTolerant`

### Service.config.location.LocationType

Indicates the type of the location requested. Must of one of the following: `CURRENT` `LAST` `CURRENT_OR_LAST` `INITIAL`.

- String: `CURRENT`

### service.standard.location.MinimumAcceptableAccuracy

Indicates the minimum possible value for an acceptable accuracy. The value is the radius in meters.

- Type: Integer (100)
- Value: `1 to MAX_INT`

### service.standard.location.MinimumAccuracy

Indicates minimum possible value for the requested accuracy. The value is the radius in meters.

- Type: Integer (100)
- Value: `1 to MAX_INT`

### service.config.MaximumTargets

Indicates the maximum number of addresses a group can contain. This is used in the getLocationForGroup operation, and when the ProcessUptoLimit is set to `true`.

- Type: Integer
- Value: 5

### service.standard.MaximumNotificationAddresses

Indicates the maximum number of addresses for which a notification can be setup.

- Type: Integer
- Value: 10

### service.config.ProcessUpToLimit

Indicates whether to process the first range of targets within the MaximumTargets (getLocationForGroup) or the MaximumNotificationAddress (Geographical & PeriodicNotification) range.

- Type: Boolean
- Value: `False`

### service.standard.MaximumNotificationDuration

The maximum amount of time in milliseconds that it takes to set up a notification.

- Long: 3600000
- Value: `1 to MAX_LONG`

### service.standard.MaximumNotificationFrequency

The maximum rate of notification deliveries in milliseconds.

- Long: 60000
- Value: `1 to MAX_LONG`

### service.standard.UnlimitedCountAllowed

Indicates whether clients may request an unlimited number of notifications.

- Type: Boolean
- Value: `True`

### service.standard.MaximumNotificationCount

Indicates the maximum number of notifications that may be requested.

- Type: Integer (10)
- Value: `1 to MAX_INT`

### service.standard.location.MinimumTrackingAccuracy

Indicates the minimum possible value for tracking accuracy. The value is the radius in meters.

- Type: Integer (100)
- Value: `1 to MAX_INT`

### service.standard.location.GeographicalNotificationAvailable

Indicates if notifications are set on a geography.

- Type: Boolean
- Value: `True`

### service.standard.PeriodicNotificationAvailable

Indicates if a periodic notification be set up.

- Type: Boolean
- Value: `True`

### service.standard.location.MinimumRequestedAccuracy

Indicates minimum possible value for requested accuracy. The value is the radius in meters Used in the startPeriodicNotification operation.

- Type: Integer (100)
- Value: `1 to MAX_INT`

### notification.Endpoint.Type

Determines the type of import component to which your application connects. If the value is set to OneAPI, the message is routed to the callout corresponding to the TL_OneAPIImport. If the value is set to JSON, the message is routed to the callout corresponding to TL_JSONImport. If the value is null, then the application uses the import component that is defined for the Web service your application is associating with.

- Type: string
- Value: null

## Default service policies for Parlay X Multimedia Messaging over MM7

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Multimedia Messaging over MM7 in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console .

The following polices are configured through the Service Policy Manager. These policies are expected to be in the SOAP header. They may be stored elsewhere, however.

### service.config.ProcessUpToLimit

If true, then the service will enforce the `service.config.MaximumTargets` policy.

- Type: Boolean
- Value: `True`

### service.config.MaximumTargets

The maximum number of targets the service should use in a single message. This number is used if `ProcessUpToLimit` is set to **true**.

- Type: Integer
- Value: `5`

### service.config.messaging.target.Ranges

The list of target URI ranges separated by semicolons.

- Type: SemiColonStringList
- Value: not applicable

### service.config.messaging.target.Aliases

The list of valid alias names currently configured with target ranges, separated by semicolons.

- Type: SemiColonStringList

- Value: not applicable

### service.config.messaging.ConfirmDelivery

This value is required. It relates to the MMS code on the backend, which communicates the extra tracking information and reports back to MMS the delivery result. This delays the final status for a send request. The backend administers a report on each target. The `ReceiptRequest` is a request for notification results from the MMS send request. This requires the policy to be true, which notifies the event being submitted and validates the notification.

- Type: Boolean
- Value: `False`

### service.config.StatusRetainTime

The number of minutes to keep the send data status records in the database.

- Type: Integer
- Value: `43200` (30 days)

### service.config.ReceivedRetainTime

The number of minutes to keep the received data messages in the database.

- Type: Integer
- Value: 43200 (30 days)

### service.standard.ChargingSupported

If true, charging information may be supplied in the Web service request. If false, charging information should not be supplied in the Web service request.

- Type: Boolean
- Value: `False`

### service.custom.PriorityEnabled

If true, `service.custom.RequestPriority` must be configured.

- Type: Boolean
- Value: `False`

### service.custom.RequestPriority

A priority number used with `service.custom.Priority`.

- Type: Integer
- Value: not applicable

### service.custom.Priority

Prefix used to build the following:

> <*service.custom.Priority*> .alias. <*service.custom.Request.priority*> .Primary
> *service.custom.priority*> .alias. <*service.custom.RequestPriority*> .Secondary

- Type: String
- Value: not applicable

### service.custom.ChargeClassEnabled

If true, `service.custom.ChargeClass` must be configured.

- Type: Boolean
- Value: `False`

### service.custom.ChargeClass

Value for the service code in `MM7SubmitReq`.

- Type: String
- Value: not applicable

# Default policies for Parlay-based Web service implementations

Telecom Web Services Server supports Web service implementations that operate over Parlay 3.x and 4.x APIs with Parlay 5.0 Multimedia Messaging (MMM).

## Default service policies for Parlay X Terminal Status over Parlay

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Terminal Status over Parlay in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Terminal Status over Parlay parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by prefixing it with `service.config` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

### service.standard.BusyAvailable

Defines whether `busy` can be returned as a status or be a trigger.

- Type: Boolean
- Value: `true`

### service.standard.MaximumNotificationFrequency

Maximum rate of notification delivery in milliseconds (also can be considered minimum time between notifications).

- Type: Integer
- Value: 1

### service.standard.MaximumNotificationDuration

Maximum amount of time a notification can be configured for. Smaller values are more restrictive.

- Type: Integer
- Value: `3600000` (1 hour)

### service.standard.MaximumNotificationCount

Maximum number of notifications that can be requested. Smaller values are more restrictive.

- Type: Integer

- Value: `10`

### service.standard.MaximumNotificationAddresses

Maximum number of addresses for which a notification can be set up. Smaller values are more restrictive.

- Type: Integer
- Value: `50`

### service.standard.UnlimitedCountAllowed

Defines whether clients can request an unlimited number of notifications.

- Type: Boolean
- Value: `true`

### service.standard.GroupSupport

Defines whether groups can be included with addresses.

- Type: Boolean
- Value: `true`

### service.standard.NestedGroupSupport

Defines whether nested groups can be included in group definitions.

- Type: Boolean
- Value: `true`

## Default service policies for Parlay X Terminal Location over Parlay

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to the Parlay X Terminal Location over Parlay in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Terminal Location over Parlay parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by prefixing it with `service.config.TerminalLocation_IMS.` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

### service.config.location.AltitudeRequested

Determines whether altitude information should be requested when location requests are made.

- Type: Boolean
- Value: equals value of `AltitudeSometimesAvailable` (ignored if `UseExtentedLocation=false`)

### service.standard.location.AltitudeSometimesAvailable

Determines whether altitude information should be requested when location requests are made if `AltitudeRequested` is not specified.

- Type: Boolean

- Value: `false` (ignored if UseExtentedLocation=false)

### service.standard.location.AltitudeAlwaysAvailable

Included per standard, but is always ignored and does not effect the system in any way.

- Type: Boolean
- Value: `false`

### service.config.location.LocationPriority

Indicates the location priority parameter to be passed to the Parlay gateway. Must be either `P_M_NORMAL` or `P_M_HIGH`.

- Type: Integer
- Value: `P_M_NORMAL` (ignored if `UseExtentedLocation=false`)

### service.standard.location.MinimumAcceptableAccuracy

Indicates minimum possible value, in meters, for acceptable accuracy.

- Type: Integer
- Value: `50` (range: `1` to `2^31 - 1`)

### service.standard.location.MinimumAccuracy

Indicates minimum possible value, in meters, for requested accuracy.

- Type: Integer
- Value: `50` (range: `1` to `2^31 - 1`)

### notification.Endpoint.Type

Determines the type of import component to which your application connects. If the value is set to OneAPI, the message is routed to the callout corresponding to the TL_OneAPIImport. If the value is set to JSON, the message is routed to the callout corresponding to TL_JSONImport. If the value is null, then the application uses the import component that is defined for the Web service your application is associating with.

- Type: string
- Value: null

## Default service policies for Parlay X SMS over Parlay

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X SMS over Parlay in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X SMS over Parlay parameters found in the TWSS Administration Console can be supplied on a per-requester or per-operation level by using the prefix `service.config.SMS_IMS.` and including the resulting header within the incoming SOAP header. Policy-based setting overrides only allow policies that are more restrictive than the settings found in the console.

### service.standard.VirtualNumber

The number that will be used as source address in Parlay sendSms call. In Parlay, this needs to be valid address in the network; otherwise the sendSms() call will fail. Required.

- Type: string
- Value: `tel:111112223300`

### service.standard.MaximumNotificationAddresses

Maximum number of addresses for which a notification can be set up. For some services, this limit is used only if `service.config.ProcessUpToLimit` is true. Not required.

- Type: integer
- Value: `5`

### service.config.ChargeForFirstSegmentOnly

If true, then charging information for large messages is included only in the first segment. If false, then charging information is included in all segments. (In the case of a single-segment message, charging information is always included in the segment.) Not required.

- Type: Boolean
- Value: `true`

### service.config.ChargingCode

Charging code to be passed to the Parlay gateway along with the request, for billing purposes. Not required.

- Type: string
- Value: null

### service.config.messaging.ConfirmDelivery

The SMS code informs the backend to track and report the delivery results. The final send request results will be delayed until the backend reports back. Not required for SMPP, but required for Parlay.

- Type: Boolean
- Value: `false`

### service.config.MaximumTargets

The maximum number of targets the Web service implementation should use in a single message. This number is used by the common layer if `service.config.ProcessUpToLimit` is set to true. Not required.

- Type: integer
- Value: `5`

### service.config.ProcessUpToLimit

If true, then enforce `service.config.MaximumTargets`. Not required.

- Type: Boolean

- Value: `true`

### service.config.ReceivedRetainTime

Time in minutes to keep the received data messages in the database. Not required.
- Type: integer
- Value: `43200` (30 days)

### service.config.StatusRetainTime

Time in minutes to keep the send data status records in the database. Not required.
- Type: integer
- Value: `43200` (30 days)

### service.config.messaging.target.Aliases

The list of valid alias names currently configured, with target ranges separated by semicolons. Not required.
- Type: SemiColonStringList
- Value: null

### service.config.messaging.target.Ranges

The list of target URI ranges separated by semicolons. Not required.
- Type: SemiColonStringList
- Value: null

### service.custom.priorityEnabled

Whether priority is enabled.
- Type: Boolean
- Value: `true`

### service.custom.requestPriority

The priority value that is set.
- Type: integer
- Value: `1`

### notification.Endpoint.Type

Determines the type of import component to which your application connects. If the value is set to OneAPI, the message is routed to the callout corresponding to the SMS_OneAPIImport. If the value is set to JSON, the message is routed to the callout corresponding to SMS_JSONImport. If the value is null, then the application uses the import component that is defined for the Web service your application is associating with.
- Type: string
- Value: null

## Default service policies for Parlay X Call Handling over Parlay

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X Call Handling over Parlay in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The following service policies can be established for Parlay X Call Handling over Parlay Mbean configuration attributes.

### service.config.callhandling.AudioContentAvailable

Accepts audio content for playing with the audio player.
- Type: Boolean
- Value: `True`

### service.config.callhandling.AudioFormatsSupported

A comma-separated string of audio formats (for example WAV, MP3, AU).
- Type: String
- Value: `WAV`

### service.config.callhandling.TextToSpeechAvailable

Accepts text as an input for processing text-to-speech.
- Type: Boolean
- Value: `True`

### service.config.callhandling.VoiceMailAvailable

Indicates whether voice mail is available.
- Type: Boolean
- Value: `True`

### service.config.callhandling.VoiceXmlAvailable

Accepts VoiceXML for processing with a VoiceXML browser.
- Type: Boolean
- Value: `True`

### service.config.standardGroupSupport

Indicates whether groups may be included with addresses.
- Type: Boolean
- Value: `False`

### service.config.NestedGroupSupport

Indicates whether nested groups are supported in group definitions.
- Type: Boolean
- Value: `False`

## Default service policies for Parlay X Call Notification over Parlay

There are no default service polices for Parlay X Call Notification over Parlay.

### Default service policies for Parlay X Third Party Call over Parlay

Policy attributes are retrieved from Subscription Management in the Access Gateway and are passed to Parlay X Third Party Call over Parlay in the SOAP headers. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

The Parlay X Third Party Call over Parlay policies will be configured through the Policy Management component. The policies are assumed to be in the SOAP header and retrieved in the Web controller layer. If the policies are not found in the TWSS SOAP header, then the Web service uses the values that are defined in the TWSS Administration Console.

#### service.config.StatusRetainTime

The time in minutes that it takes to retain status after the call terminates.

- Type: Integer
- Value: `43200` (30 days)

#### service.standard.ChargingSupported

Indicates whether charging is supported for makeCall operations.

- Type: Boolean
- Value: `False`

## Changed policy names for migration

If you are migrating your Telecom Web Services Server (TWSS) system from version 6.1.1, the names of several default policies have changed.

Refer to the lists for the names of default policies in TWSS version 6.1.1 and in the current version.

Click the name of a Web service implementation to display detailed descriptions of the policies that are associated with it.

**Parlay X Call Notification over SIP/IMS**

No default policies.

**Parlay X Presence over SIP/IMS**

*Table 54. Default policies for Parlay X Presence over SIP/IMS*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.config.Presence_IMS.PresenceServerSIPURI | service.config.presence.PresenceServerSIPURI |
| service.config.Presence_IMS.PresenceServerResourceName | service.config.presence.PresenceServerResourceName |
| service.config.Presence_IMS.ServiceImplementationName | service.config.ServiceImplementationName |
| not applicable | service.config.presence.PresenceFetchTimeoutInMillis |
| service.config.Presence_IMS.SubscribePresenceTimeout | service.config.presence.SubscribePresenceTimeout |
| service.config.Presence_IMS.MaximumNotificationFrequency | service.standard.MaximumNotificationFrequency |
| service.config.Presence_IMS.DefaultNotificationDuration | service.config.DefaultNotificationDuration |
| service.config.Presence_IMS.MaximumCount | service.standard.MaximumCount |
| not applicable | service.standard.UnlimitedCountAllowed |

*Table 54. Default policies for Parlay X Presence over SIP/IMS (continued)*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.config.Presence_IMS..GroupSupport | service.standard.GroupSupport |
| service.config.Presence_IMS.NestedGroupSupport | service.standard.NestedGroupSupport |
| service.config.Presence_IMS.InterOperatorIdentifier | service.config.ims.InterOperatorIdentifier |
| not applicable | service.config.AdditionalGroupURISchemes |

## Parlay X Terminal Status over SIP/IMS

*Table 55. Default policies for Parlay X Terminal Status over SIP/IMS*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.config.TerminalStatus_IMS.PresenceServerSIPURI | service.config.presence.PresenceServerSIPURI |
| service.config.TerminalStatus_IMS.PresenceServerResourceName | service.config.presence.PresenceServerResourceName |
| service.config.TerminalStatus_IMS.ServiceImplementationName | service.config.ServiceImplementationName |
| service.config.TerminalStatus_IMS.TerminalBusyRpidActivities | service.config.presence.TerminalBusyRpidActivities |
| service.config.TerminalStatus_IMS.SIPFetchTimeoutInMillis | service.config.presence.PresenceFetchTimeoutInMillis |
| service.config.TerminalStatus_IMS.BusyAvailable | service.standard.BusyAvailable |
| service.config.TerminalStatus_IMS.MaximumNotificationFrequency | service.standard.MaximumNotificationFrequency |
| service.config.TerminalStatus_IMS.MaximumNotificationDuration | service.standard.MaximumNotificationDuration |
| not applicable | service.config.DefaultNotificationDuration |
| service.config.TerminalStatus_IMS.MaximumCount | service.standard.MaximumCount |
| service.config.TerminalStatus_IMS.UnlimitedCountAllowed | service.standard.UnlimitedCountAllowed |
| service.config.TerminalStatus_IMS..GroupSupport | service.standard.GroupSupport |
| service.config.TerminalStatus_IMS.NestedGroupSupport | service.standard.NestedGroupSupport |
| service.config.TerminalStatus_IMS.InterOperatorIdentifier | service.config.ims.InterOperatorIdentifier |
| not applicable | service.config.AdditionalGroupURISchemes |
| not applicable | service.standard.MaximumNotificationAddresses |

## Parlay X Third Party Call over SIP/IMS

*Table 56. Default policies for Parlay X Third Party Call over SIP/IMS*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.config.ThirdPartyCall_IMS.interOperatorIdentifier | service.config.ims.InterOperatorIdentifier |
| service.config.ThirdPartyCall_IMS.mediaServer | service.config.sip.MediaServerURI |
| service.config.ThirdPartyCall_IMS.pHeaderEnabled | service.config.ims.PHeaderEnabled |
| service.config.ThirdPartyCall_IMS.trafficEstimatePerRequest | service.config.sip.GeneratedTrafficEstimate |
| not applicable | service.config.sip.OutboundProxy |
| not applicable | service.config.sip.OutboundProxyResourceName |
| not applicable | service.config.StatusRetainTime |
| service.config.ThirdPartyCall_IMS.callSessionControlServer | not applicable |
| falseservice.config.ThirdPartyCall_IMS.SIPProxyResourceSpec | not applicable |

### Parlay X Payment (PostPaid)

*Table 57. Default policies for Parlay X Payment (PostPaid)*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.config.Payment_PostPaid_IMS.PaymentUsageRecordEnabled | service.config.PaymentUsageRecordEnabled |
| service.config.Payment_PostPaid_IMS.ResultsUsageRecordEnable | service.config.ResultsUsageRecordEnable |
| service.config.Payment_PostPaid_IMS.CEIEnabled | service.config.CEIEnabled |
| service.config.Payment_PostPaid_IMS.Currency | service.config.Currency |

### Parlay X Address List Manager over XCAP

Supported for TWSS versions 6.2 and 7.0 only.

### WAP Push over SMPP

Supported for TWSS versions 6.2 and 7.0 only.

### Parlay X SMS over SMPP

*Table 58. Default policies for Parlay X SMS over SMPP*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.standard.VirtualNumber | service.standard.VirtualNumber |
| not applicable | service.standard.MaximumNotificationAddresses |
| not applicable | service.config.ChargeForFirstSegmentOnly |
| service.config.ConfirmDelivery | service.config.messaging.ConfirmDelivery |
| service.standard.MaximumTargets | service.config.MaximumTargets |
| service.common.ProcessUpToLimit | service.config.ProcessUpToLimit |
| service.config.ReceivedRetainTime | service.config.ReceivedRetainTime |
| service.config.StatusRetainTime | service.config.StatusRetainTime |
| service.common.Target.Aliases | service.config.messaging.target.Aliases |
| service.common.Target.Ranges | service.config.messaging.target.Ranges |
| not applicable | service.custom.messaging.ServiceType |
| not applicable | service.custom.priorityEnabled |
| not applicable | service.custom.requestPriority |

### Parlay X Terminal Location over MLP

Supported for TWSS versions 6.2 and 7.0 only.

### Parlay X Multimedia Messaging over MM7

Supported for TWSS versions 6.2 and 7.0 only.

### Parlay X Terminal Status over Parlay

Supported for TWSS versions 6.2 and 7.0 only.

### Parlay X Terminal Location over Parlay

*Table 59. Default policies for Parlay X Terminal Location over Parlay*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.config.AltitudeRequested | service.config.location.AltitudeRequested |
| service.standard.AltitudeSometimesAvailable | service.standard.location.AltitudeSometimesAvailable |
| service.standard.AltitudeAlwaysAvailable | service.standard.location.AltitudeAlwaysAvailable |
| service.config.LocationPriority | service.config.location.LocationPriority |
| service.standard.MinimumAcceptableAccuracy | service.standard.location.MinimumAcceptableAccuracy |
| service.standard.MinimumAccuracy | service.standard.location.MinimumAccuracy |

### Parlay X SMS over Parlay

*Table 60. Default policies for Parlay X SMS over Parlay*

| Policy name in TWSS version 6.1.1 | Policy name in TWSS version 6.2 and 7.0 |
|---|---|
| service.standard.VirtualNumber | service.standard.VirtualNumber |
| not applicable | service.standard.MaximumNotificationAddresses |
| not applicable | service.config.ChargeForFirstSegmentOnly |
| not applicable | service.config.ChargingCode |
| service.config.ConfirmDelivery | service.config.messaging.ConfirmDelivery |
| service.standard.MaximumTargets | service.config.MaximumTargets |
| service.common.ProcessUpToLimit | service.config.ProcessUpToLimit |
| service.config.ReceivedRetainTime | service.config.ReceivedRetainTime |
| service.config.StatusRetainTime | service.config.StatusRetainTime |
| service.common.Target.Aliases | service.config.messaging.target.Aliases |
| service.common.Target.Ranges | service.config.messaging.target.Ranges |
| not applicable | service.custom.priorityEnabled |
| not applicable | service.custom.requestPriority |

### Parlay X Call Handling over Parlay

Supported for TWSS version 7.0 only.

### Parlay X Call Notification over Parlay

No default policies.

### Parlay X Third Party Call over Parlay

Supported for TWSS version 7.0 only.

## Reference: usage records for the Web service implementations

TWSS supports usage records for all of the Web service implementations.

# Usage records for Web service implementations based on SIP/IMS

Telecom Web Services Server supports Web service implementations that operate over SIP/IMS.

## Usage records for Parlay X Call Notification over SIP/IMS

Parlay X Call Notification over SIP/IMS uses the service usage record to store network events, caller information, and called party information.

Each Telecom Web Services Server Web service implementation generates a service usage record that describes how the service was used for accounting and billing purposes. Service usage records are stored in relational table format. Each service usage record contains common event data that can be used to unique identify the service record, and that references a properties table containing application-specific attributes. This provides a uniform infrastructure for creating and storing service usage records.

Service usage records consist of general service usage information that may be generated at multiple points during service execution. An event type field is used to differentiate the different recording points. Each service implementation defines the event types (generation points), status codes, and service data attributes that are generated for storage in the service usage table. The Parlay X Call Notification over SIP/IMS creates a service record by calling the Usage Record component Web service.

### Service data for CallNotificationManager operation

The following table lists part names and service attributes used by the CallNotificationManager.

*Table 61. startCallNotification and stopCallNotification service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `SERVICE` | Identifies the service implementation that is writing the usage record |
| `EVENTTYPE` | Identifies the place in the service implementation where the service record is recorded |
| `CODE` | One of the following:<br>`0 = Operation successful`<br>`1 = General failure` |
| | Name requesting the operation |
| `REQUESTER` | |
| `METHOD` | Identifies the method writing the usage record. If there is an error, a brief failure reason will be attached to the operation and separated by ":" (for example; `setAddressBean:RemoteException`) |
| `CORRELATOR` | Correlator value sent with the request |
| `CALLED_PARTY` | Not available for: `stopCallNotification` |

*Table 61. startCallNotification and stopCallNotification service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| CALL_EVENTS | A string of combination network events from:<br>called_number<br>busy<br>no_answer<br>not_reachable |

## Service data for CallNotification operation

The following table lists part names and service attributes used by CallNotification operation.

*Table 62. CallNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| SERVICE | Identifies the service implementation that is writing the usage record |
| EVENTTYPE | Identifies the place in the service implementation where the service record is recorded |
| CODE | One of the following:<br>0 = Operation successful<br>1 = General failure |
| | |
| CALL_NOTIFICATION | One of the following:<br>Sending = Ready to call PX Notification Delivery Web service<br>Error = Invoking PX Notification Delivery Web service error |
| | |
| CORRELATOR | The correlator value is sent with the entity bean |
| CALLER | Converted to a string |
| CALLED_PARTY | Converted to a string |
| NETWORK_EVENT | The network event that triggered the notification.<br>It can be one of the following:<br>NotifyCalledNumber<br>NotifyBusy<br>NotifyNotReachable<br>NotifyNoAnswer |

## Usage records for Parlay X Presence over SIP/IMS

The service usage database is used by all services, including Parlay X Presence over SIP/IMS, to record events related to a service request. Parlay X Presence over SIP/IMS creates a Presence service usage record by calling the Usage Record component Web service

## Presence service implementation records based on incoming service requests

These usage records capture Parlay X-based API invocations from external clients. A single event covers marking the beginning and end of request processing.

The following table lists part names and service attributes used for subscribePresence.

*Table 63. subscribePresence service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | Copied from Access Gateway - supplied headers |
| `APPLICATION_OF_SERVICE` | The data which describes the application of the watcher |
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `NOTIFICATION_ENDPOINT` | Web service endpoint used for callbacks |
| `PRESENCE_ATTRIBUTES` | Lists all PresenceAttributeTypes if subscribePresenceRequest. The attributes are empty which indicates a wildcard value. |
| `TARGETS` | Pulled from notifySubscriptionRequest.Presentity part |
| `EXPANDED_TARGETS` | Group List Management Server - Targets can have group URIs, EXPANDED_TARGETS list containing single entries for each user/terminal |
| `EXPANDED_TARGETS_COUNT` | Total number of targets for which status has to be queried |
| `RESULT` | Contains status information in the case of a successful operation, otherwise it contains an exception type |
| `ROOT_CAUSE` | Only present if an exception occurred because of unrecoverable internal error |

The following table lists part names and service attributes used for getUserPresence.

*Table 64. getUserPresence service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | Copied from Access Gateway-supplied headers |
| `TARGETS` | URI Addresses of the terminal(s) whose status have to be retrieved |
| `PRESENCE_ACTIVITY` | Not present if the Activity is not requested in the getUserPresenceRequest.Attributes. Not present if an exception is returned to the requester |
| `PRESENCE_PLACE` | Not present if the Place is not requested in the getUserPresenceRequest.Attributes. Not present if an exception is returned to the requester |
| `PRESENCE_PRIVACY` | Not present if the Privacy is not requested in the getUserPresenceRequest.Attributes. Not present if an exception is returned to the requester |
| `PRESENCE_SPHERE` | Not present if the Sphere is not requested in the getUserPresenceRequest.Attributes. Not present if an exception is returned to the requester |
| `PRESENCE_CONTACTS` | Not present if the Communication is not requested in the getUserPresenceRequest.Attributes. Not present if an exception is returned to the requester |
| `PRESENCE_CONTACT_PRIORITIES` | Not present if the Communication is not requested in the getUserPresenceRequest.Attributes. Corresponds to the values in ResultContacts. Not present if an exception is returned to the requester |

*Table 64. getUserPresence service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `RESULT` | Only present if an exception is returned for the requester |
| `ROOT_CAUSE` | Only present if an exception occurred due to an unrecoverable internal error |

The following table lists part names and service attributes used for startPresenceNotification.

*Table 65. startPresenceNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTOR` | Copied from Access Gateway-supplied headers |
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `NOTIFICATION_ENDPOINT` | Web service endpoint used for callbacks |
| `PRESENCE_ATTRIBUTES` | Lists all of the PresenceAttributeTypes if subscribePresenceRequest .Attributes was empty (indicating a wildcard value) |
| `TARGETS` | URI Addresses of the terminal(s) whose status have to be retrieved |
| `EXPANDED_TARGETS` | Group List Management server - TARGETS can have group URIs, EXPANDED_TARGETS list which contains a single entry for each user/terminal |
| `EXPANDED_TARGETS_COUNT` | Total number of targets for which status has to be queried |
| `FREQUENCY` | Copied from incoming request; in milliseconds |
| `NOTIFICATION_EXPIRES_AT` | Copied from incoming request, converted to milliseconds since the epoch |
| `NOTIFICATION_COUNT` | Identifies maximum number of notifications to deliver |
| `CHECK_IMMEDIATE` | Indicates that the service is being asked that notification be given immediately after the subscription setup completes |
| `RESULT` | The comma separated list of presentity addresses for which the subscriber could not subscribe to the requested attributes, or the returned an exception |
| `ROOT_CAUSE` | Only present if the exception occurred because of an unrecoverable internal error |

The following table lists part names and service attributes used for endPresenceNotification.

*Table 66. endPresenceNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTOR` | Identifies the application that issues the service request |
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `RESULT` | Only present if an exception was returned to the requester |
| `ROOT_CAUSE` | Only present if the exception occurred because of an unrecoverable internal error |

The following table lists part names and service attributes used for notifySubscription_DeliveryAttempted.

Table 67. notifySubscriptionDeliveryAttempted service attributes and descriptions

| Attribute | Description |
|---|---|
| CORRELATOR | Used to communicate with endpoint for callbacks |
| TARGET | Pulled from the notifySubscriptionRequest.Presentity part |
| PRESENCE_PERMISSIONS | An xsd:Boolean indicating if the requester was authorized to access the Presentity in Address |
| APP_CORRELATION_ID | Used to match this DeliveryAttempted record with a DeliveryResult |

The following table lists part names and service attributes used for notifySubscription_DeliveryResult.

Table 68. notifySubscriptionDeliveryResult service attributes and descriptions

| Attribute | Description |
|---|---|
| FAILURE_REASON | Empty if the DeliveryStatus = Success |
| FAILURE_DETAIL | Empty if the DeliveryStatus = Success |
| APP_CORRELATION_ID | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for subscriptionEnded_DeliveryAttempted.

Table 69. subscriptionEndedDeliveryAttempted service attributes and descriptions

| Attribute | Description |
|---|---|
| TARGET | Pulled from notifySubscriptionRequest.Presentity part |
| REASON | Describes why the subscription has ended - For example; Timeout or Blocked |
| APP_CORRELATION_ID | Used to match this DeliveryAttempted record with a DeliveryResult |

The following table lists part names and service attributes used for subscriptionEnded_DeliveryResult.

Table 70. subscriptionEndedDeliveryResult service attributes and descriptions

| Attribute | Description |
|---|---|
| FAILURE_REASON | Empty if the DeliveryStatus = Success |
| FAILURE_DETAIL | Empty if the DeliveryStatus = Success |
| APP_CORRELATION_ID | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for publish.

Table 71. publish service attributes and descriptions

| Attribute | Description |
|---|---|
| REQUESTER | Identifies the application that issues the service request |
| PRESENCE_ACTIVITY | Activity value, obtained from publishReq.getAttribute |
| PRESENCE_PLACE | Obtained from publishRequest.Attributes; if not specified, the value is PlaceNone |

*Table 71. publish service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| PRESENCE_PRIVACY | Obtained from publishRequest.Attributes; if not specified, the value is PrivacyNone |
| PRESENCE_SPHERE | Obtained from publishRequest.Attributes; if not specified, the value is SphereNone |
| PRESENCE_CONTACTS | Comma separated list of the Communication Means Contact values obtained from publishReq.getAttribute |
| PRESENCE_CONTACT_PRIORITIES | Comma separated list of the Communication Means Priority values obtained from publishReq.getAttribute |
| RESULT | Present if an exception is returned to the requester |
| ROOT_CAUSE | Present if an exception occurs due to an unrecoverable internal error |

## Usage records for Parlay X Terminal Status over SIP/IMS

The service usage database is used by all services, including Parlay X Terminal Status over SIP/IMS, to record events related to a service request. The payment service invokes the Usage Record component Web service to write service usage records to the database

### Parlay X Terminal Status over SIP/IMS records based on incoming service requests

These usage records capture Parlay X-based API invocations from external clients. A single event covers marking the beginning and end of request processing.

The following table lists part names and service attributes used for getStatus.

*Table 72. getStatus service attributes and descriptions*

| Attribute | Description |
|---|---|
| REQUESTER | Requester name requesting this operation - Identifies the application that issues the service request |
| TARGETS | URI Addresses of the terminal (s) whose status have to be retrieved |
| RESULT | Contains status information in case of successful operation, otherwise contains an exception type |
| ROOT_CAUSE | Only present if an exception occurs due to an unrecoverable internal error |

The following table lists part names and service attributes used for getStatusForGroup.

*Table 73. getStatusForGroup service attributes and descriptions*

| Attribute | Description |
|---|---|
| REQUESTER | Requester name requesting this operation |
| TARGETS | URI Addresses of the terminal(s) whose status have to be retrieved |

*Table 73. getStatusForGroup service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `EXPANDED_TARGETS` | Group List Management server - TARGETS can have group URIs, EXPANDED_TARGETS list contains a single entry for each user/terminal. |
| `EXPANDED_TARGETS_COUNT` | Total number of targets for which status has to be queried |
| `RESULT` | Contains status information in case of successful operation, otherwise, contains an exception type |
| `ROOT_CAUSE` | Present only if an exception occurred because of an unrecoverable internal error |

The following table lists part names and service attributes used for startNotification.

*Table 74. startNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | Identifies the application that issued the service request |
| `TARGETS` | URI Addresses of the terminal(s) whose status have to be retrieved |
| `EXPANDED_TARGETS` | Group List Management server - TARGETS can have group URIs, EXPANDED_TARGETS list contains a single entry for each user/terminal |
| `EXPANDED_TARGETS_COUNT` | Total number of targets for which status has to be queried |
| `CORRELATOR` | Endpoint used for callbacks |
| `NOTIFICATION_ENDPOINT` | An endpoint URI used for notifications by the service |
| `NOTIFICATION_CRITERIA` | When an application receives a notification |
| `CHECK_IMMEDIATE` | Identifies if an immediate notification should be returned. Has to be one of the following:<br>`Yes`<br>`No` |
| `FREQUENCY` | Duration between sending two separate asynchronous notifications |
| `NOTIFICATION_EXPIRES_AT` | Expiration time for the notification |
| `NOTIFICATION_COUNT` | Maximum number of notifications delivered - either this value does not exist or it is a value of zero |

The following table lists part names and service attributes used for endNotification.

Table 75. endNotification service attributes and descriptions

| Attribute | Description |
|---|---|
| REQUESTER | Identifies the application that issues the service request |
| CORRELATOR | Used to communicate with endpoint for callbacks |
| RESULT | Contains status information in case of successful operation, otherwise, contains an exception type |
| ROOT_CAUSE | Present only if an Exception occurred because of unrecoverable internal error |

## Parlay X Terminal Status over SIP/IMS records based on outgoing service notifications

These usage records capture outbound Web service-based notifications/callbacks related to StartNotification operations. Global transaction IDs allow these to be correlated with related StartNotification events.

The following table lists part names and service attributes used for statusNotification_DeliveryAttempted.

Table 76. statusNotification_DeliveryAttempted service attributes and descriptions

| Attribute | Description |
|---|---|
| CORRELATOR | Used to communicate with endpoint for callbacks |
| TARGETS | Formatted target address |
| DELIVERY_STATUS | One of the PresenceAttributeType enumeration status values or the returned Exception |
| APP_CORRELATION_ID | DeliveryResult |

The following table lists part names and service attributes used for statusNotification_DeliveryResult.

Table 77. statusNotification_DeliveryResult service attributes and descriptions

| Attribute | Description |
|---|---|
| FAILURE_REASON | Empty if the DeliveryStatus = Success |
| FAILURE_DETAIL | Empty if the DeliveryStatus = Success |
| APP_CORRELATION_ID | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for statusChanged_DeliveryAttempted.

Table 78. statusChangedDeliveryAttempted service attributes and descriptions

| Attribute | Description |
|---|---|
| CORRELATOR | Used to communicate with endpoint for callbacks |
| TARGETS | Pulled from the notifySubscriptionRequest.Presentity part |

Table 78. statusChangedDeliveryAttempted service attributes and descriptions (continued)

| Attribute | Description |
|---|---|
| APP_CORRELATION_ID | Used to match this DeliveryAttempted record with a DeliveryResult |
| PRESENCE_ACTIVITY | Not present if the *Activity* is not in the ChangedAttributes |
| PRESENCE_PLACE | Not present if the *Place* is not in the ChangedAttributes |
| PRESENCE_PRIVACY | Not present if the *Privacy* is not in the ChangedAttributes |
| PRESENCE_SPHERE | Not present if the *Sphere* is not in the ChangedAttributes |
| PRESENCE_CONTACTS | Not present if the *Communication* is not in the ChangedAttributes |
| PRESENCE_CONTACT_PRIORITIES | Not present if the *Communication* is not in the ChangedAttributes - Corresponds to the values in PRESENCE_CONTACTS |

The following table lists part names and service attributes used for statusChanged_DeliveryResult.

Table 79. statusChangedDeliveryResult service attributes and descriptions

| Attribute | Description |
|---|---|
| FAILURE_REASON | Empty if the DeliveryStatus = Success |
| FAILURE_DETAIL | Empty if the DeliveryStatus = Success |
| APP_CORRELATION_ID | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for statusError_DeliveryAttempted.

Table 80. statusError_DeliveryAttempted service attributes and descriptions

| Attribute | Description |
|---|---|
| CORRELATOR | Used to communicate with endpoint for callbacks |
| TARGET | May be empty |
| MESSAGE_ID | May be either a fault message for the single request or a data item for a group response |
| REASON | Describes why the subscription ended - For example; Timeout or Blocked |
| APP_CORRELATION_ID | Used to match this DeliveryAttempted record with a DeliveryResult |

The following table lists part names and service attributes used for statusError_DeliveryResult.

Table 81. statusError_DeliveryResult service attributes and descriptions

| Attribute | Description |
|---|---|
| FAILURE_REASON | Empty if the DeliveryStatus = Success |
| FAILURE_DETAIL | Empty if the DeliveryStatus = Success |
| APP_CORRELATION_ID | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for statusEnd_DeliveryAttempted.

*Table 82. statusEnd_DeliveryAttempted service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `APP_CORRELATION_ID` | Used to match this DeliveryAttempted record with a DeliveryResult |

The following table lists part names and service attributes used for statusEnd_DeliveryResult.

*Table 83. statusEnd_DeliveryResult service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `FAILURE_REASON` | Empty if the DeliveryStatus = Success |
| `FAILURE_DETAIL` | Empty if the DeliveryStatus = Success |
| `APP_CORRELATION_ID` | Used to match this DeliveryResult record with a DeliveryAttempted |

## Usage records for Parlay X Call Handling over SIP/IMS

Parlay X Call Handling over SIP/IMS calls the Usage Record component Web service to record events related to a service request.

The following table lists the information that is passed to the Usage Record component Web service for Call Handling.

*Table 84. Parlay X Call Handling over SIP/IMS service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `REQUESTER` | The request name |
| `SERVICE` | The actual service name |
| `TARGET` | The target string after the validation has processed |
| `START_TIME` | The actual start time |
| `RESPONSE_TIME` | Time it takes to process the response, in milliseconds |
| `ERROR_TYPE` | The error type of the request. Has to be one of the following:<br>`ServiceException`<br>`PolicyException` |
| `FAILURE_REASON` | The error message identifier of the request. Has to be one of the following:<br>`SVC xxxx`<br>`POL xxxx` |
| `FAILURE_DETAIL` | The error text of the request |
| `ADDRESS_STATUS` | Validating the status of a single target |
| `TARGET_STATUS` | Validating the status of a specific target |
| `TARGET_SUCCESSFUL` | The success of a specific target |

*Table 84. Parlay X Call Handling over SIP/IMS service attributes and descriptions (continued)*

| **Attribute** | **Description** |
|---|---|
| `TARGET_ERROR_TYPE` | The error type of a specific target. Has to be one of the following:<br>`PolicyException`<br>`ServiceException` |
| `TARGET_FAILURE_REASON` | The error message ID of a specific target. Has to be one of the following:<br>`SVC xxxx`<br>`POL xxxx` |
| `TARGET_FAILURE_DETAIL` | The error text of a specific target |

## Usage records for Parlay X Third Party Call over SIP/IMS

The Parlay X Third Party Call over SIP/IMS uses the Usage Record component Web service to record events related to a service request.

### Third Party Call service implementation service record

Parlay X Third Party Call over SIP/IMS usage records are created by calling the Usage Record component Web service, which creates entries in the `USAGERECORDS` table.

The following table lists part names and service attributes used for Third Party Call.

*Table 85. Third Party Call service implementation attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `service` | Generated by the Access Gateway and passed in the `globalTransactionID` SOAP header. |
| `eventType` | Identifies the event for this record.<br>Has to be one of the following:<br>`StartCall`<br>`EndCall` |
| | |
| `code` | One of the following:<br>`0 = Operation Successful`<br>`1 = General failure`<br>`2 = Failure due to policy or privacy` |
| | |
| | |
| `REQUESTER` | SIP user originating the call |
| `CALLER` | SIP user being called |
| `CALLED_PARTY` | The requester URI is converted to a string that identifies the application that issued the service request |

Table 85. Third Party Call service implementation attributes and descriptions (continued)

| Attribute | Description |
|---|---|
| `CHARGE_DESCRIPTION` | Descriptive text to used for information and billing |
| `CHARGE_CURRENCY` | Currency identifier as defined in ISO 4217 |
| `CHARGE_AMOUNT` | Amount to be charged |
| `CHARGE_CODE` | Charging code, referencing a contract under which the charge is applied |
| `CALL_STATUS` | Status of the call at the current time |
| `START_TIME` | Time the call began |
| `CALL_DURATION` | Duration of the call |
| `TERMINATION_CAUSE` | Cause of the termination of the call |

## Usage records for Web service implementations based on IBM WebSphere software

Telecom Web Services Server (TWSS) supports Web service implementations that interoperate with other components within the IBM WebSphere product group–including WebSphere Application Server, IBM XDMS, and other instances of TWSS.

### Usage records for Parlay X Payment (PostPaid)

The Parlay X Payment (PostPaid) Web service implementation uses usage records to store accounting and billing information.

The Payment service implementation generates Payment service usage records for use by security and accounting tools. All Payment service implementation usage records contain the requester URI to identify which client application sent the Payment service request, the operation name to identify the requested action, and the reference code to uniquely identify the Payment request.

The Payment service issues a WriteUsageRecordRequest to the Usage Record component Web service at the following event points:

- To write a Payment service charge record to the service usage database (if configured to do so in the service.config.PaymentUsageRecordEnabled policy).
- To record the results of the request before the Payment service terminates (if configured to do so in the service.config.ResultsUsageRecordEnabled policy).

If the Usage Record component Web service cannot write a Payment service usage record in the service usage database, a WriteUsageRecordFault fault is generated.

An exception or a WriteUsageRecordFault fault that is generated while trying to create a Payment service charge record is a severe error. The Payment service calls the Fault and Alarm component Web service with the fault information and a SVC0270 (charge failed) fault is returned to the Payment Web service requester.

An exception or a WriteUsageRecordFault fault that is generated while trying to create a Payment service results record is not a severe error and does not cause a fault to be returned to the Payment Web service requester. The Payment service calls the Fault and Alarm component Web service to record the exception or fault information and continue.

## Payment service usage records

The service usage database is used by all services, including the Payment service implementation, to record events related to a service request. The Payment service invokes the Usage Record component Web service to write service usage records to the database, and also uses the database to store charging information for use by a billing mediator application. Storage of charging information in the service usage database is dependent on the configuration of the service.config.PaymentUsageRecordEnabled policy attribute. Depending on the billing mediator application implementation, Payment service usage records might be deleted in the service usage database after being used to successfully integrate charging information to a user's account in the billing domain.

## Payment service charge record

The Payment service implementation creates a Payment service charge record by calling the Usage Record component Web service. The following tables describe the information that is passed to the Usage Record component Web service in order to create a Payment service charge record.

The following table lists part names and service attributes used by payment service.

*Table 86. payment service charge service attributes and descriptions*

| Attribute | Description |
|---|---|
| `globalTransactionId` | Generated by Access Gateway and passed in the `globalTransactionID` SOAP header |
| `SERVICE` | The service implementation that is writing the usage record.<br>This record is created if `requester.operation.postpaid.PaymentUsageRecordEnabled` is set to *TRUE* |
| `eventType` | The operation name. This record is created if `service.config.Payment_PostPaid_IMS.PaymentUsageRecordEnabled` is set to *TRUE* |
| `code` | Indicates that the Payment service implementation is proceeding without errors |
| `REQUESTOR` | The requester URI is converted to a string that identifies the application that issued the service request |
| `OPERATION` | The name of the operation that was invoked.<br>Can be any of the following:<br>`chargeAmount`<br>`refundAmount`<br>`chargeVolume`<br>`refundVolume` |
| `REFERENCE_CODE` | This is the reference code from the input message - This code uniquely identifies the Payment request (in case of customer payment dispute) The reference code is not the same things as the charging code |
| `END_USER_IDENTIFIER` | The account for the end user who will be billed is converted from xsd:anyURI to a String - The content format of the URI data is operator service dependent |

The following table lists part names and service attributes used by chargeAmount and refundAmount.

*Table 87. chargeAmount and refundAmount service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CHARGE_DESCRIPTION` | Description text to be used for information and billing text |
| `CHARGE_CURRENCY` | **Optional**: Currency identifier as defined in ISO 4217 |
| `CHARGE_AMOUNT` | **Optional**: (if CHARGE_CODE is not empty) Amount to be charged converted from xsd:Decimal to a string |

*Table 87. chargeAmount and refundAmount service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `CHARGE_CODE` | **Optional**: (if CHARGE_AMOUNT is not empty) References a contract under which the charge is applied - The Code field in the ChargingInformation part may be empty because it is an optional item |

The following table lists part names and service attributes used by chargeVolume and refundVolume.

*Table 88. chargeVolume and refundVolume service attributes and descriptions*

| Attribute | Description |
|---|---|
| `VOLUME` | The volume to be charged converted from xsd:long to a String |
| `BILLING_TEXT` | Textual information to appear on the bill |
| `UNIT` | The unit used for measuring volume (for example, bytes) |
| `CONTRACT` | The number of a contract that may govern use |
| `SERVICE` | The name of the service to be used (such as SendMultimediaMessage) |
| `OPERATION` | The name of the operation to be used |

**Note:** The number of property attributes for `UNIT`, `CONTRACT`, `SERVICE`, and `OPERATION` will vary, and may be zero. These attributes are used to perform rating operations.

## Payment service results record

The Usage Record component Web service creates entries in the `USAGERECORDS` table. The following tables describe the information that is passed to the Usage Record component Web service in order to create a Payment service results record.

The following table lists part names and service attributes used by payment service results.

*Table 89. payment service results attributes and descriptions*

| Attribute | Description |
|---|---|
| `globalTransactionId` | Generated by Access Gateway and passed in the `globalTransactionID` SOAP header |
| `SERVICE` | This is a Payment service charge record -<br>This record is created at the end of the Payment service |
| `eventType` | This identifies a Payment service charge record. This record is created at the end of the Payment service; if service.config.Payment_PostPaid_IMS.ResultsUsageRecordEnabled is set to *TRUE* |
| `code` | One of the following:<br>`0 = Payment service request was successful`<br>`1 = Payment service request was a general failure` |
| | |
| `REQUESTER` | This is the requester URI converted to a String; it identifies the application that issued the Payment service request |
| `OPERATION` | The name of the operation that was invoked.<br>Can be any of the following:<br>`chargeAmount`<br>`refundAmount`<br>`chargeVolume`<br>`refundVolume` |
| `REFERENCE_CODE` | The reference code from the input message - The code uniquely identifies the Payment request (in case of customer Payment dispute) The reference code is not the same thing as the charging code |

## Usage records for Parlay X Address List Manager over XCAP

The Parlay X Address List Manager over XCAP Web service uses the service usage record to record any event generated during create, read, update and delete group definition stored within XDMS.

For each operation, the following utility classes are invoked.

The following table lists part names and service attributes used by the Address List Manager service.

*Table 90. Address List Manager service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `Transaction ID` | Generated by the ESB and stored in the SOAP Headers will serve this purpose |
| `service` | The service implementation that is writing the usage record |
| `eventType` | The operation name, for example *addMember* |
| `code` | One of the following:<br>`0 = Operation successful`<br>`1 = General failure`<br>`2 = Failure due to policy / TWSS Service Platform component` |
| | |
| | |
| `Requester` | The requester URI converted to a string – identifies the application that issued the service request |
| `INPUT_URI` | The XCAP URI used |
| `OUTPUT_MESSAGE_NUMBER` | The number of group members , for operations that return multiple members |

# Usage records for Web service implementations based on Direct Connect protocols

Telecom Web Services Server supports Web service implementations that operate over Direct Connect protocols such as SMPP, MLP, and MM7.

## Usage records for WAP Push over SMPP

The WAP Push over SMPP Web service implementation uses the Usage Record component Web service to record usage information which includes exception details. For the synchronous part of the Web service request, an error condition maps to an exception which is then returned to the caller.

### WAP Push over SMPP service record

Usage records for WAP Push over SMPP are created by calling the Usage Record component Web service, which creates entries in the `USAGERECORDS` table.

The following table describes the information that is passed to the Usage Record component Web service for this Web service implementation.

*Table 91. Usage record component service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | The time that the usage record was written |
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for WAP Push operation

The following table lists part names and service attributes used by the WAP Push operation.

*Table 92. WAP Push service attributes and descriptions*

| Attribute | Description |
|---|---|
| `TARGET` | Formatted target address |
| `MESSAGE_LENGTH` | Length of the message in bytes |
| `CHARGE_CODE` | Optional: references the contract a charge is applied to |
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester |
| `REQUEST_IDENTIFIER` | Identifies a specific WAP Push delivery request |
| `REGISTRATION_IDENTIFIER` | The registration identifier for the particular request |

*Table 92. WAP Push service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| DELIVERY_STATUS | Successfully delivered to Terminal - the final status |
| | Unsuccessful delivery; the message could not be delivered before it expired - the final status |
| | Unable to provide delivery receipt notification. When this state is stored, a ServiceException will also be returned. Therefore this state must be set by the front end and no queued request will take place - the final status |
| | Successful delivery to network - not the final status |
| | Delivery status unknown because it was handed off to another network. This will be the case when the network has reported that it is unable to deliver the message because the device is not available - not the final status |
| | The message is still queued for delivery - a temporary state, pending transition to one of the preceding states |
| SESSION_ID | A session ID for the requester |
| ASSIGNMENT_ID | The assignment ID associated with sending the message |
| SERVICE_ACTIVATION_NUMBER | The target address; must be unique |
| REFERENCE | The reference code from the input message - uniquely identifies the request |
| ENDPOINT | Endpoint URI that will receive the notifications |
| INTERFACE_NAME | Name of the WAP Push interface used for this request |
| REQUESTER | Requester name requesting this operation |
| CONFIRM_DELIVERY | A confirmation message returned when the message is delivered. Has to be one of the following:<br>`Yes`<br>`No` |
| NETWORK_ID | Name of the network |

## Service data for service exceptions

The following table lists part names and service attributes used by service exceptions.

*Table 93. Exception data service attributes and descriptions*

| Attribute | Description |
|---|---|
| FAILURE_REASON | Error message identifier indicating the error |
| FAILURE_DETAIL | A detailed error text message |
| ERROR_TYPE | Type of error encountered |

## WAP Push over SMPP service record

The Usage Record component Web service create entries in the `USAGERECORDS` table. The following table describes the information that is passed to the Usage Record component Web service in order to create a service record for WAP Push over SMPP.

The following table lists part names and service attributes used by WAP Push.

Table 94. WAP Push service attributes and descriptions

| Attribute | Description |
|---|---|
| `SOAM6002` | An error occurred while creating the records in the WAPPUSHSENDDATA table |
| `SOAM6003` | An error occurred while reading the records in the WAPPUSHSENDDATA table |
| `SOAM6004` | A Database error occurred while creating the records in the WAPPUSHSENDDATA table |
| `SOAM6005` | No records were found in the WAPPUSHSENDDATA table for the transactionid |
| `SOAM6006` | The send message has failed due to the delivery confirmation request was not able to be setup |
| `SOAM6007` | The send message has failed due to a problem creating the notification record. Check the error string returned for more information |
| `SOAM6020` | The target address value is null |
| `SOAM6021` | One of targets address value is an empty string |
| `SOAM6022` | The message parameter is null |
| `SOAM6023` | The message parameter is an empty string |
| `SOAM6025` | Problem in the database layer of WAPPUSH. |
| `SOAM6026` | The request identifier parameter is null |
| `SOAM6027` | The request identifier parameter is an empty string |
| `SOAM6028` | The value of request identifier parameter is not known |
| `SOAM6029` | The value of delivery status return from the gateway - this error should never happen unless the gateway returns undefined type of delivery status |
| `SOAM6030` | Requester ID value in the TWSS context (MdsContext) is null |

*Table 94. WAP Push service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `SOAM6031` | TWSS context (MdsContext) value is null |
| `SOAM6032` | Error when creating notification record for *sendWAPPushSI* request |
| `SOAM6055` | This should never happen unless there is coding error |
| `SOAM6059` | The service enable MBean in the administration console is set to *false* |
| `SOAM6060` | The service policy *service.standard.ChargingSupported* is set to *false* and charging information is passed in as part of the parameter |
| `SOAM6076` | There is a problem with the database connection |
| `SOAM6077` | There is a unknown problem |
| `SOAM6079` | There is a problem with updating the status on *sendWAPPushSI* record |
| `SOAM6080` | A problem with finding the segment status for the transaction |
| `SOAM6085` | This happens after the Parlay call to *startNotification* on the gateway is successful and database update call is made |
| `SOAM6086` | This happens at the time that the notification from the gateway message has been received by the specified device |
| `SOAM6090` | This is a database problem which occurs during the purging process |
| `SOAM6091` | This is a database problem which occurs during the purging process |
| `SOAM6100` | Database service is not up and running or database authentication failed |
| `SOAM6101` | The href parameter is null. |
| `SOAM6102` | The href parameter is an empty string |
| `SOAM6103` | The action parameter is null |

*Table 94. WAP Push service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `SOAM6104` | The action parameter is an empty string or an incorrect value. Valid values are signal-none, signal-low, signal-medium, signal-high, delete. |

### Direct Connect (SMPP) service record

The Usage Record component Web service create entries in the `USAGERECORDS` table. The following table lists part names and service attributes used by Direct Connect (SMPP).

*Table 95. Direct Connect (SMPP) service attributes and descriptions*

| Attribute | Description |
|---|---|
| `SOAM6150` | The service policy *service.common.messaging.target.Aliases* is not in the TWSS context (MdsContext) |
| `SOAM6151` | The service policy *service.config.ConfirmDelivery* is not in the TWSS context (MdsContext) |
| `SOAM6152` | No alias names were defined in the administration console |
| `SOAM6153` | A problem with getting the MBean value that was set in the administration console |
| `SOAM6154` | Unable to connect to the specified SMSC Server |
| `SOAM6155` | Internal Error with the *SmppSessionManager* |
| `SOAM6156` | An error occurs while sending the message to the SMSC |
| `SOAM6157` | Failure to send the Enquire Link Response message |
| `SOAM6158` | Failure to send the Deliver SM Response message |
| `SOAM6159` | Failure to send the Generic Non-Acknowledge message |

## Usage records for Parlay X SMS over SMPP

Parlay X SMS over SMPP uses the Usage Record component Web service to record events related to a service request.

### Parlay X SMS over SMPP service record

Usage records for Parlay X SMS over SMPP are created by calling the Usage Record component Web service, which creates entries in the `USAGERECORDS` table.

The following table describes the information that is passed to the Usage Record component Web service for this Web service implementation.

*Table 96. Usage record component service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | The time that the usage record was written |
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for sendSms, sendSmsLogo, and sendSmsRingtone operations

The following table lists part names and service attributes used by sendSms, sendSmsLogo, and sendSmsRingtone. One usage record is written per target (without delivery confirmation).

*Table 97. sendSms, sendSmsLogo, and sendSmsRingtone service attributes and descriptions*

| Attribute | Description |
|---|---|
| `DELIVERY_STATUS` | Successfully delivered to Terminal. This is a final status. |
|  | Unsuccessful delivery; the message could not be delivered before it expired. This is a final status. |
|  | Cannot provide delivery receipt notification. A notifySmsDeliveryReceipt operation will produce `DeliveryNotificationNotSupported` to indicate that delivery receipt for the specified address in a sendSms request is not supported. When this state is stored, a ServiceException is be returned. Therefore this state must be set by the front end, and no queued request will take place. This is the final status. |
|  | Successful delivery to network. This is not the final status. |
|  | Delivery status unknown because it was handed off to another network. This will be the case when the network has reported that it is unable to deliver the message because the device is not available. This is not the final status. |
|  | The message is still queued for delivery. This is a temporary state, pending transition to one of the preceding states. |
| `START_TIME` | Time the operation began |
| `REQUESTER` | Value of the requester ID |
| `TARGET` | The target address for this message |
| `SERVICE` | The service name requesting this operation |
| `NETWORK_ID` | Name of the network |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |

## Service data for getSmsDeliveryStatus operation

The following table lists part names and service attributes used by getSmsDeliveryStatus. One usage record is written per operation.

*Table 98. getSmsDeliveryStatus service attributes and descriptions*

| Attribute | Description |
|---|---|
| `START_TIME` | Time of the beginning of the operation |
| `SERVICE` | The service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |
| `REQUEST_IDENTIFIER` | Identifies a specific SMS delivery request |

## Service data for getReceivedSms operation

The following table lists part names and service attributes used by getReceivedSms. One usage record is written per operation.

*Table 99. getReceivedSms service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REGISTRATION_IDENTIFIER` | The registration identifier for the particular request. |
| `START_TIME` | Time of the beginning of the operation. |
| `SERVICE` | Service name requesting this operation. |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request. |

## Service data for startSmsNotification operation

The following table lists part names and service attributes used by startSmsNotification. One usage record is written per operation.

*Table 100. startSmsNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `START_TIME` | Time the operation began |
| `SERVICE_ACTIVATION_NUMBER` | The target address known as the *smsServiceActivationNumber*; must be unique |
| `SERVICE` | The service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |

## Service data for stopSmsNotification operation

The following table lists part names and service attributes used by startSmsNotification. One usage record is written per operation.

*Table 101. stopSmsNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `SERVICE` | Service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |

### Service data for service exceptions

The following table lists part names and service attributes used for service exceptions.

*Table 102. Service data service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_DETAIL` | A detailed error message |
| `FAILURE_REASON` | An error message ID indicating the nature of the error |

### Service data for notifySmsDeliveryReceipt

Usage records are not generated for the notifySmsDeliveryReceipt operation.

## Usage records for Parlay X Terminal Location over MLP

The Parlay X Terminal Location over MLP uses the Usage Record component Web service to record events related to a service request.

### Parlay X Terminal Location over MLP service record

Usage records for Parlay X Terminal Location over MLP are created by calling the Usage Record component Web service, which creates entries in the USAGERECORDS table.

The following table lists part names and service attributes used by Usage Record component Web service.

*Table 103. Usage record service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | When the usage record was written |

*Table 103. Usage record service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for getLocation, getTerminalDistance and getLocationForGroup operations

The following table lists part names and service attributes used by getLocation, getTerminalDistance, and getLocationForGroup.

*Table 104. getLocation, getTerminalDistance, and getLocationForGroup service attributes and descriptions*

| Attribute | Description |
|---|---|
| `START_TIME` | Time the operation began |
| `REQUEST_ACCEPTABLE_ACCURACY` | Lowest accuracy acceptable to requester |
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>`Yes`<br>`No` |
| `REQUESTED_DELAY_TOLERANCE` | Priority of response time versus accuracy. Has to be one of the following:<br>`DELAY_TOLERANT`<br>`LOW_DELAY` |
| `REQUESTER` | Requester name requesting this operation |
| `RESULT_ACCURACY` | Accuracy of the resulting location |
| `REQUESTED_RESPONSE_TIME` | Maximum time that the application can accept to wait for a response |

*Table 104. getLocation, getTerminalDistance, and getLocationForGroup service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `IS_USING_EXTENDED_REQUEST` | Whether extended location requests should be made. Has to be one of the following:<br>`Yes`<br>`No` |
| `REQUESTED_MAXIMUM_AGE_OF_RESPONSE` | Maximum acceptable age of the location information that is returned |
| `TARGET` | Formatted target address |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `REQUESTED_ACCURACY` | Accuracy of location information requested |
| `LOCATION_TYPE` | Parlay gateway location type strings. Has to be one of the following:<br>`CURRENT_OR_LAST`<br>`CURRENT`<br>`INITIAL`<br>`LAST` |
| `REQUEST_PRIORITY` | Location priority parameter passed to the MLP location server. Has to be one of the following:<br>`NORMAL`<br>`HIGH` |

## Service data for startGeographicalNotification operation

The following table lists part names and service attributes used by startGeographicalNotification. One usage record is written per target.

*Table 105. startGeographicalNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | Requester name requesting this operation |
| `FREQUENCY` | Minimum time between notifications |
| `DURATION` | Maximum duration of a notification request |
| `NOTIFICATION_EXPIRES_AT` | Expiration time of the notification |
| `TARGET` | Formatted target address |
| `NOTIFICATION_CRITERIA` | Indicates whether the notification should occur when the terminal enters or leaves the target area. Has to be one of the following:<br>`LEAVING`<br>`ENTERING` |

*Table 105. startGeographicalNotification service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `LOCATION_TYPE` | MLP location type strings. Has to be one of the following:<br>`CURRENT_OR_LAST`<br>`CURRENT`<br>`INITIAL`<br>`LAST` |
| `REQUEST_PRIORITY` | Location priority parameter passed to the MLP Location server. Has to be one of the following:<br>`NORMAL`<br>`HIGH` |
| `START_TIME` | Time the operation began |
| `NOTIFICATION_TRIGGER_RADIUS` | Requested triggered radius |
| `NOTIFICATION_ENDPOINT` | Endpoint URL that will receive the notifications |
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>`Yes`<br>`No` |
| `RESPONSE_TIME` | Amount of time it take for the response |
| `NOTIFICATION_COUNT` | Number of notifications requested |
| `REQUEST_IDENTIFIER` | Unique identifier generated by the MLP Location Server corresponding to a notification |

## Service data for startPeriodicNotification operation

The following table lists part names and service attributes used by startPeriodicNotification. One usage record is written per target.

*Table 106. startPeriodicNotifications service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `FREQUENCY` | Frequency of a notification request |

*Table 106. startPeriodicNotifications service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `REQUEST_PRIORITY` | Location priority parameter passed to the MLP location server. Has to be one of the following:<br>`NORMAL`<br>`HIGH` |
| `TARGET` | Formatted target address |
| `REQUESTER` | Requester name requesting this operation |
| `NOTIFICATION_EXPIRES_AT` | Expiration time of the notification |
| `START_TIME` | Time the operation began |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `DURATION` | Maximum duration of a notification request |
| `NOTIFICATION_ENDPOINT` | Endpoint URL that will receive the notifications |
| `LOCATION_TYPE` | Location type strings sent to the MLP Location server. Has to be one of the following:<br>`CURRENT_OR_LAST`<br>`CURRENT`<br>`INITIAL`<br>`LAST` |
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>`Yes`<br>`No` |

## Service data for endNotification operation

The following table lists part names and service attributes used by endNotification. One usage record is written per operation.

*Table 107. endNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | Requester name requesting this operation |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `START_TIME` | Time the operation began |
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `TRANSACTIONID` | Unique for the transaction |

Table 107. endNotification service attributes and descriptions (continued)

| Attribute | Description |
|---|---|
| `REQUESTID` | Unique for the request |

### Service data for service exceptions: All operations

The following table lists part names and service attributes used for service exceptions.

Table 108. Exception service attributes and descriptions

| Attribute | Description |
|---|---|
| `FAILURE_REASON` | Error message identifier indicating the error |
| `FAILURE_DETAIL` | A detailed error text message |
| `ERROR_TYPE` | Type of error encountered |

### Service data for locationNotification, locationError, locationEnd

Usage records are not generated for the locationNotification, locationError, and locationEnd operations.

## Usage records for Parlay X Multimedia Messaging over MM7

The Parlay X Multimedia Messaging over MM7 Web service implementation uses the Usage Record component Web service to get, send, and start notification messages.

### Parlay X Multimedia Messaging over MM7 service record

Usage records for Parlay X Multimedia Messaging over MM7 are created by calling the Usage Record component Web service, which creates entries in the `USAGERECORDS` table.

The following table describes the information that is passed to the Usage Record component Web service for this Web service implementation.

Table 109. Usage record component service attributes and descriptions

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | The time that the usage record was written |

Table 109. Usage record component service attributes and descriptions (continued)

| Attribute | Description |
|---|---|
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for sendMessage operation

The following table lists part names and service attributes used by sendMessage. One usage record is written per target (without delivery confirmation).

Table 110. sendMessage service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |
| `START_TIME` | The start time for the request for example, *yyyy-MM-dd'T'HH:mm:ssZ* |

## Service data for sendMessage Delivery Status operation

The following table lists part names and service attributes used by sendMessage to report delivery status. One usage record is written per operation.

Table 111. sendMessage Delivery Status service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |

Table 111. sendMessage Delivery Status service attributes and descriptions (continued)

| Attribute | Description |
|---|---|
| `DELIVERY_STATUS` | Delivery status - possible values are:<br>`DeliveredToTerminal`<br>`DeliveryImpossible`<br>`DeliveryNotificationNotSupported`<br>`DeliveredToNetwork`<br>`DeliveryUncertain`<br>`MessageWaiting` |
| `TARGET` | The target, for example the `tel:` prefix followed by a telephone number |

The following table lists part names and service attributes used by getMessageDeliveryStatus.

Table 112. getMessageDeliveryStatus service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |
| `REQUEST_IDENTIFIER` | The *requestIdentifier* value that was included in the request |

## Service data for startMessageNotification operation

The following table lists part names and service attributes used by getReceivedMessages.

Table 113. getReceivedMessages service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |

The following table lists part names and service attributes used by getMessage.

Table 114. getMessage service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |

The following table lists part names and service attributes used by startMessageNotification. One usage record is written per operation.

Table 115. startMessageNotification service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |

Table 115. startMessageNotification service attributes and descriptions (continued)

| Attribute | Description |
|---|---|
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |
| `START_TIME` | The start time for the request for example, *yyyy-MM-dd'T'HH:mm:ssZ* |
| `CORRELATOR` | Correlation information from the end point - Used to communicate with endpoint for callbacks, this information must be unique for this service and requester |
| `SERVICE_ACTIVATION_NUMBER` | The service activation number, for example a telephone number |
| `NOTIFICATION_ENDPOINT` | An endpoint URI used for notifications by the service |

### Service data for stopMessageNotification operation

The following table lists part names and service attributes used by stopMessageNotification. One usage record is written per operation.

Table 116. stopMessageNotification service attributes and descriptions

| Attribute | Description |
|---|---|
| `SERVICE` | The service implementation that is writing the usage record |
| `eventTYPE` | The operation name |
| `REQUESTER` | The originator of the request |
| `CORRELATOR` | Correlation information from the end point - Used to communicate with endpoint for callbacks, this information must be unique for this service and requester |

### Service data for notifyMessageReception, notifyMessageDeliveryReceipt

Usage records are not generated for the notifyMessageReception and notifyMessageDeliveryReceipt operations.

# Usage records for Parlay-based Web service implementations

Telecom Web Services Server supports Web service implementations that operate over Parlay 3.x and 4.x APIs with Parlay 5.0 Multimedia Messaging (MMM).

## Usage records for Parlay X Terminal Status over Parlay

Parlay X Terminal Status over Parlay uses the Usage Record component Web service to record Web service invocation details.

### Parlay X Terminal Status over Parlay service record

Usage records for Parlay X Terminal Status over Parlay are created by capturing Parlay X-based API invocations from external clients. A single event marks the beginning and the end of request processing.

The following table describes the information that is passed to the Usage Record component Web service for this Web service implementation.

*Table 117. Usage record component service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | The time that the usage record was written |
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for getStatus operation

The following table lists part names and service attributes used by getStatus.

*Table 118. getStatus service attributes and descriptions*

| Attribute | Description |
|---|---|
| `ASSIGNMENT_ID` | The Parlay assignment ID associated with sending the message |
| `REQUESTER` | Requester name requesting this operation |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `START_TIME` | Time of the beginning of the operation |
| `SERVICE` | Service name requesting this operation |
| `TARGET` | Formatted target address |

## Service data for getStatusForGroup operation

The following table lists part names and service attributes used by getStatusForGroup.

*Table 119. getStatusForGroup service attributes and descriptions*

| Attribute | Description |
|---|---|
| ASSIGNMENT_ID | The Parlay assignment ID associated with sending the message |
| REQUESTER | Names of the requesters for this operation: a group of terminal addresses, separated by comas |
| RESPONSE_TIME | Amount of time to take for the response |
| START_TIME | Time of the beginning of the operation |
| SERVICE | Service name requesting this operation |
| TARGET | Formatted target address |

## Service data for startNotification operation

The following table lists part names and service attributes used by startNotification.

*Table 120. startNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| REQUESTER | Requester name requesting this operation |
| SERVICE | Service name requesting this operation |
| DURATION | Maximum duration of a notification request |
| NOTIFICATION_EXPIRES_AT | Expiration time of the notification |
| TARGET | Formatted target address |
| ASSIGNMENT_ID | The Parlay assignment ID associated with sending the message |
| RESPONSE_TIME | Amount of time to take for the response |
| START_TIME | Time of the beginning of the operation |
| CORRELATOR | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| NOTIFICATION_ENDPOINT | Endpoint URL that will receive the notifications |
| FREQUENCY | Frequency of a notification request |
| NOTIFICATION LIMIT | Notification limit |

## Service data for endNotification operation

The following table lists part names and service attributes used by endNotification.

*Table 121. endNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| ASSIGNMENT_ID | The Parlay assignment ID associated with sending the message |
| REQUESTER | Requester name requesting this operation |
| RESPONSE_TIME | Amount of time to take for the response |
| START_TIME | Time of the beginning of the operation |

*Table 121. endNotification service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `SERVICE` | Service name requesting this operation |

## Service data for service exceptions

The following table lists part names and service attributes used for service exceptions.

*Table 122. Exception service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_REASON` | Error message identifier indicating the error |
| `FAILURE_DETAIL` | A detailed error text message |
| `ERROR_TYPE` | Type of error encountered |

## Parlay X Terminal Status over Parlay records based on outgoing service notifications

These usage records capture outbound Web service-based notifications/callbacks related to StartNotification operations. Global transaction IDs allow these to be correlated with related StartNotification events.

The following table lists part names and service attributes used for statusNotification_DeliveryAttempted.

*Table 123. statusNotification_DeliveryAttempted service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `TARGETS` | Formatted target address |
| `DELIVERY_STATUS` | One of the PresenceAttributeType enumeration status values or the returned Exception |
| `APP_CORRELATION_ID` | DeliveryResult |

The following table lists part names and service attributes used for statusNotification_DeliveryResult.

*Table 124. statusNotification_DeliveryResult service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_REASON` | Empty if the DeliveryStatus = Success |
| `FAILURE_DETAIL` | Empty if the DeliveryStatus = Success |
| `APP_CORRELATION_ID` | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for statusError_DeliveryAttempted.

*Table 125. statusError_DeliveryAttempted service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `TARGET` | May be empty |
| `MESSAGE_ID` | May be either a fault message for the single request or a data item for a group response |
| `REASON` | When an application receives the notification |
| `APP_CORRELATION_ID` | Used to match this DeliveryAttempted record with a DeliveryResult |

The following table lists part names and service attributes used for statusError_DeliveryResult.

*Table 126. statusError_DeliveryResult service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_REASON` | Empty if the DeliveryStatus = Success |
| `FAILURE_DETAIL` | Empty if the DeliveryStatus = Success |
| `APP_CORRELATION_ID` | Used to match this DeliveryResult record with a DeliveryAttempted |

The following table lists part names and service attributes used for statusEnd_DeliveryAttempted.

*Table 127. statusEnd_DeliveryAttempted service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Used to communicate with endpoint for callbacks |
| `APP_CORRELATION_ID` | Used to match this DeliveryAttempted record with a DeliveryResult |

The following table lists part names and service attributes used for statusEnd_DeliveryResult.

*Table 128. statusEnd_DeliveryResult service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_REASON` | Empty if the DeliveryStatus = Success |
| `FAILURE_DETAIL` | Empty if the DeliveryStatus = Success |
| `APP_CORRELATION_ID` | Used to match this DeliveryResult record with a DeliveryAttempted |

## Usage records for Parlay X Terminal Location over Parlay

The Parlay X Terminal Location over Parlay Web service implementation, uses the Usage Record component Web service to record events related to a service request.

### Parlay X Terminal Location over Parlay service record

Usage records for Parlay X Terminal Location over Parlay are created by calling the Usage Record component Web service, which creates entries in the `USAGERECORDS` table.

The following table describes the information that is passed to the Usage Record component Web service for this Web service implementation.

*Table 129. Usage record component service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | The time that the usage record was written |
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for getLocation operation

The following table lists part names and service attributes used by getLocation.

*Table 130. getLocation service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RESULT_CURRENCY` | Timestamp of the result |
| `START_TIME` | Time the operation began |
| `REQUEST_ACCEPTABLE_ACCURACY` | Lowest accuracy acceptable to requester |
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>`Yes`<br>`No` |

*Table 130. getLocation service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `REQUESTED_DELAY_TOLERANCE` | Priority of response time versus accuracy. Has to be one of the following:<br>`P_M_DELAY_TOLERANT`<br>`P_M_LOW_DELAY`<br>`P_M_NO_DELAY` |
| `REQUESTER` | Requester name requesting this operation |
| `RESULT_ACCURACY` | Accuracy of the resulting location |
| `REQUESTED_RESPONSE_TIME` | Maximum time that the application can accept to wait for a response |
| `IS_USING_EXTENDED_REQUEST` | Whether extended location requests should be made. Has to be one of the following:<br>`Yes`<br>`No` |
| `REQUESTED_MAXIMUM_AGE_OF_RESPONSE` | Maximum acceptable age of the location information that is returned |
| `SERVICE` | Service name requesting this operation |
| `TARGET` | Formatted target address |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `REQUESTED_ACCURACY` | Accuracy of location information requested |
| `LOCATION_METHOD` | Parlay gateway location method strings |
| `LOCATION_TYPE` | Parlay gateway location type strings. Has to be one of the following:<br>`P_M_CURRENT_OR_LAST_KNOWN`<br>`P_M_CURRENT`<br>`P_M_INITIAL` |
| `REQUEST_PRIORITY` | Location priority parameter passed to the Parlay gateway. Has to be one of the following:<br>`P_M_NORMAL`<br>`P_M_HIGH` |

## Service data for getLocationForGroup operation

The following table lists part names and service attributes used by getLocationForGroup.

*Table 131. getLocationForGroup service attributes and descriptions*

| Attribute | Description |
|---|---|
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>`Yes`<br>`No` |
| `REQUESTED_DELAY_TOLERANCE` | Priority of response time versus accuracy. Has to be one of the following:<br>`P_M_DELAY_TOLERANT`<br>`P_M_LOW_DELAY`<br>`P_M_NO_DELAY` |
| `REQUESTER` | Requester name requesting this operation |
| `RESULT_ACCURACY` | Accuracy of the resulting location |
| `REQUESTED_RESPONSE_TIME` | Maximum time that the application can accept to wait for a response |
| `REQUESTED_MAXIMUM_AGE_OF_RESPONSE` | Maximum acceptable age of the location information that is returned |

*Table 131. getLocationForGroup service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `IS_USING_EXTENDED_REQUEST` | Whether extended location requests should be made. Has to be one of the following:<br>`Yes`<br>`No` |
| `SERVICE` | Service name requesting this operation |
| `TARGET` | Formatted target address |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `REQUESTED_ACCURACY` | Accuracy of location information requested |
| `LOCATION_METHOD` | Parlay gateway location method strings |
| `LOCATION_TYPE` | Parlay gateway location type strings. Has to be one of the following:<br>`P_M_CURRENT_OR_LAST_KNOWN`<br>`P_M_CURRENT`<br>`P_M_INITIAL` |
| `REQUEST_PRIORITY` | Location priority parameter passed to the Parlay gateway. Has to be one of the following:<br>`P_M_NORMAL`<br>`P_M_HIGH` |
| `RESULT_CURRENCY` | Timestamp of the result |
| `START_TIME` | Time the operation began |

## Service data for getTerminalDistance operation

The following table lists part names and service attributes used by getTerminalDistance.

*Table 132. getTerminalDistance service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RESULT_ACCURACY` | Accuracy of the resulting location |
| `REQUESTED_RESPONSE_TIME` | Maximum time that the application can accept to wait for a response |
| `IS_USING_EXTENDED_REQUEST` | Whether extended location requests should be made. Has to be one of the following: Has to be one of the following:<br>`Yes`<br>`No` |
| `REQUESTED_MAXIMUM_AGE_OF_RESPONSE` | Maximum acceptable age of the location information that is returned |
| `SERVICE` | Service name requesting this operation |
| `TARGET` | Formatted target address |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `REQUESTED_ACCURACY` | Accuracy of location information requested |
| `LOCATION_METHOD` | Parlay gateway location method strings |
| `LOCATION_TYPE` | Parlay gateway location type strings. Has to be one of the following:<br>`P_M_CURRENT_OR_LAST_KNOWN`<br>`P_M_CURRENT`<br>`P_M_INITIAL` |

*Table 132. getTerminalDistance service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| REQUEST_PRIORITY | Location priority parameter passed to the Parlay gateway. Has to be one of the following:<br>P_M_NORMAL<br>P_M_HIGH |
| RESULT_CURRENCY | Timestamp of the result |
| START_TIME | Time the operation began |
| REQUEST_ACCEPTABLE_ACCURACY | Lowest accuracy acceptable to requester |
| IS_ALTITUDE_REQUESTED | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>Yes<br>No |
| REQUESTED_DELAY_TOLERANCE | Priority of response time versus accuracy. Has to be one of the following:<br>P_M_DELAY_TOLERANT<br>P_M_LOW_DELAY<br>P_M_NO_DELAY |
| REQUESTER | Requester name requesting this operation |

## Service data for startGeographicalNotification operation

The following table lists part names and service attributes used by startGeographicalNotification.

*Table 133. startGeographicalNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| REQUESTER | Requester name requesting this operation |
| RESULT_ACCURACY | Accuracy of the resulting location |
| REQUESTED_RESPONSE_TIME | Maximum time that the application can accept to wait for a response |
| IS_USING_EXTENDED_REQUEST | Whether extended location requests should be made. Has to be one of the following:<br>Yes<br>No |
| REQUESTED_MAXIMUM_AGE_OF_RESPONSE | Maximum acceptable age of the location information that is returned |
| REQUESTED_MAXIMUM_NOTIFICATION_FREQUENCY | Maximum frequency of a notification request |
| SERVICE | Service name requesting this operation |
| DURATION | Maximum duration of a notification request |
| NOTIFICATION_EXPIRES_AT | Expiration time of the notification |
| TARGET | Formatted target address |
| LOCATION_METHOD | Parlay gateway location method strings |
| NOTIFICATION_CRITERIA | Indicates whether the notification should occur when the terminal enters or leaves the target area. Has to be one of the following:<br>P_UL_LEAVING_AREA<br>P_UL_ENTERING_AREA |
| LOCATION_TYPE | Parlay gateway location type strings. Has to be one of the following:<br>P_M_CURRENT_OR_LAST_KNOWN<br>P_M_CURRENT<br>P_M_INITIAL |

*Table 133. startGeographicalNotification service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `REQUEST_PRIORITY` | Location priority parameter passed to the Parlay gateway. Has to be one of the following:<br>`P_M_NORMAL`<br>`P_M_HIGH` |
| `RESULT_CURRENCY` | Timestamp of the result |
| `START_TIME` | Time the operation began |
| `NOTIFICATION_TRIGGER_RADIUS` | Requested triggered radius |
| `NOTIFICATION_ENDPOINT` | Endpoint URL that will receive the notifications |
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `NOTIFICATION LIMIT` | Notification limit |
| `REQUEST_ACCEPTABLE_ACCURACY` | Lowest accuracy acceptable to requester |
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made. Has to be one of the following:<br>`Yes`<br>`No` |
| `REQUESTED_DELAY_TOLERANCE` | Priority of response time versus accuracy. Has to be one of the following:<br>`P_M_DELAY_TOLERANT`<br>`P_M_LOW_DELAY`<br>`P_M_NO_DELAY` |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `REQUESTED_ACCURACY` | Accuracy of location information requested |

## Service data for startPeriodicNotification operation

The following table lists part names and service attributes used by startPeriodicNotification.

*Table 134. startPeriodicNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `FREQUENCY` | Frequency of a notification request |
| `REQUEST_PRIORITY` | Location priority parameter passed to the Parlay gateway. Has to be one of the following:<br>`P_M_NORMAL`<br>`P_M_HIGH` |
| `LOCATION_METHOD` | Parlay gateway location method strings |
| `TARGET` | Formatted target address |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |
| `REQUESTER` | Requester name requesting this operation |
| `SERVICE` | Service name requesting this operation |
| `NOTIFICATION_EXPIRES_AT` | Expiration time of the notification |
| `START_TIME` | Time the operation began |

*Table 134. startPeriodicNotification service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `REQUESTED_DELAY_TOLERANCE` | Priority of response time versus accuracy. Has to be one of the following:<br>`P_M_DELAY_TOLERANT`<br>`P_M_LOW_DELAY`<br>`P_M_NO_DELAY` |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `DURATION` | Maximum duration of a notification request |
| `REQUEST_ACCEPTABLE_ACCURACY` | Lowest accuracy acceptable to requester |
| `NOTIFICATION_ENDPOINT` | Endpoint URL that will receive the notifications |
| `REQUESTED_RESPONSE_TIME` | Maximum time that the application can accept to wait for a response |
| `LOCATION_TYPE` | Parlay gateway location type strings. Has to be one of the following:<br>`P_M_CURRENT_OR_LAST_KNOWN`<br>`P_M_CURRENT`<br>`P_M_INITIAL` |
| `IS_ALTITUDE_REQUESTED` | Determines whether altitude information should be requested when location requests are made |

## Service data for endNotification operation

The following table lists part names and service attributes used by endNotification.

*Table 135. endNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | Requester name requesting this operation |
| `SERVICE` | Service name requesting this operation |
| `NOTIFICATION_TYPE` | Notification type. Has to be one of the following:<br>`Triggered`<br>`Periodic` |
| `ASSIGNMENT_ID` | Parlay assignment ID associated with the request |
| `RESPONSE_TIME` | Amount of time to take for the response |
| `START_TIME` | Time the operation began |
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |

## Service data for service exceptions

The following table lists part names and service attributes used for service exceptions.

*Table 136. Exception service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_REASON` | Error message identifier indicating the error |
| `FAILURE_DETAIL` | A detailed error text message |
| `ERROR_TYPE` | Type of error encountered |

### Service data for locationError, locationNotification, locationEnd

Usage records are not generated for locationError, locationNotification, and locationEnd operations.

## Usage records for Parlay X SMS over Parlay

The Parlay X SMS over Parlay Web service implementation uses the Usage Record component Web service to record events related to a service request.

### Parlay X SMS over Parlay service record

Usage records for Parlay X SMS over Parlay are created by calling the Usage Record component Web service, which creates entries in the `USAGERECORDS` table.

The following table describes the information that is passed to the Usage Record component Web service for this Web service implementation.

*Table 137. Usage record component service attributes and descriptions*

| Attribute | Description |
|---|---|
| `RECORDID` | A unique record identifier |
| `SEGMENT` | Segment number used for subsequent truncated records. This number is incremented each time an overflow of the `SERVICEDATA` field is encountered and a subsequent record is added.<br>Used by the SMS-based service implementations; a value of 0 is written for all others. |
| `GLOBALID` | Global transaction identifier |
| `SERVICE` | Name of the service |
| `HOST` | Name of the host on which the usage record is being recorded |
| `EVENTTYPE` | Name of the operation being performed |
| `RECORDTIME` | The time that the usage record was written |
| `STATUSCODE` | One of the following:<br>`0 = VALID`<br>`1 = FAILED_PRIVACY`<br>`2 = FAILED_ADDRESS_PLAN_VALIDATION`<br>`3 = EXCEEDED_MAX_TARGETS`<br>`4 = FAILED_PACER`<br>`5 = FAILED_CALL_LEVEL_PRIVACY`<br>`6 = FAILED_PROCESS_UP_TO_LIMIT`<br>`7 = FAILED_ADMISSION_CONTROL`<br>`8 = FAILED_SERVICE_NOT_RUNNING`<br>`9 = FAILED_JMS_ENQUEUE`<br>`10 = UNKNOWN_SCHEME`<br>`11 = FAILED_OPERATION`<br>`12 = FAILED_OTHER`<br>`13 = INVALID`<br>`14 = GROUPS_NOT_ALLOWED`<br>`15 = UNRESOLVED_GROUPS`<br>`16 = FAILED_TRAFFIC_SHAPING` |
| `SERVICEDATA` | Semicolon-delimited list of attributes described in the following tables, written in the format `key=value;key=value...` |

## Service data for sendSms operation

The following table lists part names and service attributes used by sendSms.

*Table 138. sendSMS service attributes and descriptions*

| Attribute | Description |
|---|---|
| `DELIVERY_STATUS`: indicates the delivery result of the address. | Successfully delivered to Terminal - the final status |
| | Unsuccessful delivery; the message could not be delivered before it expired - the final status |
| | Unable to provide delivery receipt notification. NotifySMSDeliveryReceipt function will provide DeliveryNotificationNotSupported to indicate that delivery receipt for the specified address in a SendSMSRequest is not supported. When this state is stored, a ServiceException will also be returned. Therefore this state must be set by the front end and no queued request will take place - the final status |
| | Successful delivery to network - not the final status |
| | Delivery status unknown because it was handed off to another network. This will be the case when the network has reported that it is unable to deliver the message because the device is not available - not the final status |
| | The message is still queued for delivery - a temporary state, pending transition to one of the preceding states |
| `SESSION_ID` | A session ID for the requester |
| `TARGETS` | The target address for this message |
| `ASSIGNMENT_ID` | The Parlay assignment ID associated with sending the message |
| `SERVICE` | The service name requesting this operation |
| `CHARGE_DESCRIPTION` | Description text to be used for information and billing text. If charging information is specified in the request then mandatory; otherwise, this is optional. |
| `CHARGE_CODE` | Optional: references the contract a charge is applied to |
| `CHARGE_AMOUNT` | Optional: amount to be charged |
| `CHARGE_CURRENCY` | Optional: currency ID, as defined in ISO 4217 |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |

## Service data for getSmsDeliveryStatus operation

The following table lists part names and service attributes used by getSmsDeliveryStatus.

*Table 139. getSmsDeliveryStatus service attributes and descriptions*

| Attribute | Description |
|---|---|
| `START_TIME` | Time of the beginning of the operation |
| `SERVICE` | The service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |
| `REQUEST_IDENTIFIER` | Identifies a specific SMS delivery request |

## Service data for getReceivedSms operation

The following table lists part names and service attributes used by getReceivedSms.

*Table 140. getReceivedSms service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REGISTRATION_IDENTIFIER` | The registration identifier for the particular request |
| `START_TIME` | Time of the beginning of the operation |
| `SERVICE` | Service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |

## Service data for startSmsNotification operation

The following table lists part names and service attributes used by startSmsNotification.

*Table 141. startSmsNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `SERVICE_ACTIVATION_NUMBER` | The target address known as the *smsServiceActivationNumber*; must be unique |
| `SERVICE` | The service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |

## Service data for stopSmsNotification operation

The following table lists part names and service attributes used by stopSmsNotification.

*Table 142. stopSmsNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| `CORRELATOR` | Correlation information from the end point. Used to communicate with endpoint for callbacks, this information must be unique for this service and requester. |
| `SERVICE` | Service name requesting this operation |
| `REQUESTER` | The requester URI, converted to a string that identifies the application that issued the service request |

## Service data for service exceptions

The following table lists part names and service attributes used for service exceptions.

*Table 143. Exception service attributes and descriptions*

| Attribute | Description |
|---|---|
| `FAILURE_DETAIL` | A detailed error message |
| `FAILURE_REASON` | An error message ID indicating the nature of the error |

### Service data for sendSmslogo, sendSmsRingTone, notifySmsReception, notifySmsDeliveryReceipt

Usage records are not generated for the sendSmslogo, sendSmsRingTone, notifySmsReception, and notifySmsDeliveryReceipt operations.

## Usage records for Parlay X Call Handling over SIP/IMS

Parlay X Call Handling over SIP/IMS calls the Usage Record component Web service to record events related to a service request.

The following table lists the information that is passed to the Usage Record component Web service for Call Handling.

*Table 144. Parlay X Call Handling over SIP/IMS service attributes and descriptions*

| Attribute | Description |
|---|---|
| `REQUESTER` | The request name |
| `SERVICE` | The actual service name |
| `TARGET` | The target string after the validation has processed |
| `START_TIME` | The actual start time |
| `RESPONSE_TIME` | Time it takes to process the response, in milliseconds |
| `ERROR_TYPE` | The error type of the request. Has to be one of the following:<br>`ServiceException`<br>`PolicyException` |
| `FAILURE_REASON` | The error message identifier of the request. Has to be one of the following:<br>`SVC xxxx`<br>`POL xxxx` |
| `FAILURE_DETAIL` | The error text of the request |
| `ADDRESS_STATUS` | Validating the status of a single target |
| `TARGET_STATUS` | Validating the status of a specific target |
| `TARGET_SUCCESSFUL` | The success of a specific target |
| `TARGET_ERROR_TYPE` | The error type of a specific target. Has to be one of the following:<br>`PolicyException`<br>`ServiceException` |
| `TARGET_FAILURE_REASON` | The error message ID of a specific target. Has to be one of the following:<br>`SVC xxxx`<br>`POL xxxx` |

*Table 144. Parlay X Call Handling over SIP/IMS service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| TARGET_FAILURE_DETAIL | The error text of a specific target |

## Usage records for Parlay X Call Notification over Parlay

Parlay X Call Notification over Parlay uses the Usage Record component Web service to record events in a application that sends Web service requests to the network mapper for the creating, registering, and reporting of network related events.

For accounting and billing purposes, each TWSS Web service implementation generates a service usage record that describes how the service was used. Service usage records are stored in relational table format. Each service usage record contains common event data that can be used to unique identify the service record, and that references a properties table containing application-specific attributes. This provides a uniform infrastructure for creating and storing service usage records.

Service usage records consist of general service usage information that may be generated at multiple points during service execution. An event type field is used to differentiate the different recording points. Each service implementation defines the event types (generation points), status codes, and service data attributes that are generated for storage in the service usage table.

Parlay X Call Notification over Parlay creates a service record by calling the Usage Record component Web service.

### Service data for CallNotificationManager

The following table lists part names and service attributes used by startCallNotification and startCallDirectionNotification:

*Table 145. startCallNotification and startCallDirectionNotification service attributes and descriptions*

| Attribute | Description |
|---|---|
| REQUESTER | The requester that invokes the operation |
| SERVICE | The name of the service registering the notification |
| TARGET | The target string after the validation is processed |
| START_TIME | The start time of the notification |
| RESPONSE_TIME | The time it takes to process the response, in milliseconds |
| NOTIFICATION_ENDPOINT | The endpoint which attempts the notification delivery |
| NOTIFICATION_CRITERIA | The *notificationCriteria* value which determines the status of the notification delivery attempt, should a network attempt occur |
| ASSIGNMENT_ID | The returned Parlay gateway assignment ID |
| CORRELATOR | The correlator used to identify the notification with the requester and service |

The following table lists part names and service attributes used by stopCallNotification and stopCallDirectionNotification:

*Table 146. stopCallNotification and stopCallDirectionNotification service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `REQUESTER` | The requester invoking the operation |
| `SERVICE` | The name of the service which registers the notification |
| `START_TIME` | The start time |
| `RESPONSE_TIME` | The time it takes to process the response, in milliseconds |
| `ASSIGNMENT_ID` | The returned Parlay gateway assignment ID |
| `CORRELATOR` | The correlator used to identify the notification with the requester and service |

The following table lists part names and service attributes used by Error attributes:

*Table 147. Error service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `FAILURE_DETAIL` | Detailed error message |
| `FAILURE_REASON` | Message which indicates the type of error that occurred. Has to be one of the following:<br>`SVC xxxx`<br>`POL xxxx` |

## Usage records for Parlay X Third Party Call over Parlay

Parlay X Third Party Call over Parlay uses the Usage Record component Web service to record events related to a service request.

### Parlay X Third Party Call over Parlay implementation service record

Parlay X Third Party Call over Parlay usage records are created by calling the Usage Record component Web service, which creates entries in the USAGERECORDS table.

The following table lists the information that is passed to the Usage Record component Web service in order to create a usage record for Parlay X Third Party Call over Parlay.

*Table 148. Parlay X Third Party Call over Parlay makeCall service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `RESPONSE_TIME` | Time of processing the response. |
| `SERVICE` | The actual service name. |
| `TARGET` | Target string after the validation processing. |
| `CALLER` | It contains the address of the first user involved in the call. |

*Table 148. Parlay X Third Party Call over Parlay makeCall service attributes and descriptions (continued)*

| Attribute | Description |
|---|---|
| `CALLED_PARTY` | It contains the address of the second user involved in the call. |
| `CALL_IDENTIFIER` | It identifies a specific call request. |
| `START_TIME` | The start time. |

*Table 149. Parlay X Third Party Call over Parlay endCall service attributes and descriptions*

| Attribute | Description |
|---|---|
| REQUESTER | The request name. |
| START_TIME | The start time. |
| CALL_IDENTIFIER | It identifies a specific call request. |
| RESPONSE_TIME | Time of processing the response. |
| SERVICE | The actual service name. |

*Table 150. Parlay X Third Party Call over Parlay cancelCall service attributes and descriptions*

| Header | Header |
|---|---|
| SERVICE | The actual service name. |
| START_TIME | The start time. |
| CALL_IDENTIFIER | It identifies a specific call request. |
| RESPONSE_TIME | Time of processing the response. |
| REQUESTER | The request name. |

*Table 151. Parlay X Third Party Call over Parlay getCallInformation service attributes and descriptions*

| Header | Header |
|---|---|
| CALL_STATUS | It identifies the status of the call. |
| SERVICE | The actual service name. |
| CALL_DURATION | Indicates the duration of the call. |
| CALL_IDENTIFIER | It identifies a specific call request. |
| START_TIME | The start time. |
| RESPONSE_TIME | Time of processing the response. |
| REQUESTER | The request name. |

*Table 152. Parlay X Third Party Call over Parlay Error Response service attributes and descriptions*

| Header | Header |
|---|---|
| ERROR_TYPE | Error type of a specific target. Has to be one of the following:<br>`ServiceException`<br>`PolicyException`<br>`ServiceError, others` |

*Table 152. Parlay X Third Party Call over Parlay Error Response service attributes and descriptions (continued)*

| **Header** | **Header** |
|---|---|
| ERROR_TEXT | Error text of a specific target. |
| SERVICE | The actual service name. |
| ERROR_MESSAGE_ID | Error message identifier of a specific target. Has to be one of the following:<br>`SVC xxxx`<br>`POL xxxx` |

## Attributes for the Notification Management component Web service

Notification Management component Web service registers notifications in the Notification management.

The following table lists part names and service attributes used for RegisterNotification.

*Table 153. startCallNotification and startCallDirectionNotification service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `globalTransactionID` | The global transaction ID generated by the TWSS Access Gateway |
| `requester` | The requester invoking the operation |
| `service` | The name of the service that is registering the notification |
| `correlator` | The correlator used to uniquely identify the notification in conjunction with the requester and service. |
| `operation` | The name of the operation that is registering the notification |
| `operationTarget` | The target URIs taken from the *notification-establishing* Web Service operation the client invoked. This is a parameter that is passed in on the startMessageNotification call. |
| `notificationPortTypeName` | The WSDL portType used by the service to delivery notifications |
| `notificationPortTypeNameSpace` | The *notificationPortTypeNamespace* value is the WSDL namespace associated with the notification WSDL interface portType. |
| `notificationCriterion` | The criterion value used by the service to determine whether or not a notification delivery attempt should occur. This is a parameter that is passed in on the *startMessageNotification* call. |
| `notificationEndPoint` | The endpoint used by the service to attempt notification delivery. This is a parameter passed in on the *startMessageNotification* call. It is included in the reference parameter. |

*Table 153. startCallNotification and startCallDirectionNotification service attributes and descriptions (continued)*

| **Attribute** | **Description** |
|---|---|
| `adminSupportEndPoint` | The endpoint that the Notification Management administrative service may use to forcibly terminate a live notification.The request URI for the *startMessageNotification* request will be parsed for the hostname and port that will be used for this endpoint. |
| `startTimeInMillis` | The time at which the notification began. |
| `endTimeInMillis` | The time at which the notification will expire or terminate, due to timeout considerations. A null *endTimeInMillis* means that no finite timeout exists. |
| `maxCountPerTarget` | The maximum number of notification delivery attempts that will be tried for a particular target in a particular notification. Null *maxCountPerTarget* value indicates that an unlimited number of delivery attempts may occur for each operationTarget within the notification. |
| `minPeriodInMillis` | The minimum elapsed time or period between any two notifications. Null minPeriodInMillis value indicates that the MMS/MM7 service will not perform any throttling on notification delivery attempts. |

The following table lists part names and service attributes used for RemoveNotification

*Table 154. RemoveNotification service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `globalTransactionID` | The global transaction ID generated by the Access Gateway |
| `requester` | The requester invoking the operation |
| `service` | The name of the service that is registering the notification |
| `correlator` | The correlator used to uniquely identify the notification in conjunction with the requester and service. |

The following table lists part names and service attributes used for NotificationAdministrationSupport.

*Table 155. TerminateNotificationRequest service attributes and descriptions*

| **Attribute** | **Description** |
|---|---|
| `adminTransactionID` | The transaction ID |
| `requester` | The requester invoking the operation |
| `correlator` | The service which registered the notification |
| `service` | The name of the service that is registering the notification |

# Chapter 9. Reference information

Information about supported standards, directory conventions, and terminology are provided as additional reference information to help you.

## Changes to this edition

Since the last edition of this information, the following changes have been made.

*Table 156. Change history for the product documentation*

| Edition | Date | Changes |
|---|---|---|
| First Edition | April 2009 | First issue of the product documentation. |

## Documentation conventions

Typographical conventions are used to make the documentation easier to understand.

The following conventions are used throughout the documentation:

- Variables are italicized. Italicized information indicates that you should substitute information from your environment for the value. For example:

  http://*host_name*:*port_number*
- Variables are used to indicate installation directories. The variable links to information with the default paths. For example:

  *`was_root`*/logs
- Images are used to indicate information specific to one operating system or database software. For example:

  Linux *`was_root`*`/installableApps/TWSS-Services`
- Values that you must type display in `monospace` font.
- User interface elements display as **boldfaced** text.
- Links to related information for each topic are provided at the bottom of the topic.

## Directory conventions

References in the documentation are for default directory locations. This topic describes the conventions in use for WebSphere Application Server Network Deployment.

### Default product locations when the root user or an administrator user installs the product

The root user or administrator is capable of registering shared products and installing into the default system-owned directories. These file paths are default locations, but you can install the products and create profiles in any directory where you have write access. Multiple installations of any of these products or components require multiple installation locations.

*was_root*

The following list shows default installation root directories for WebSphere Application Server Network Deployment:

- AIX `/usr/IBM/WebSphere/AppServer`
- Linux `/opt/IBM/WebSphere/AppServer`

*was_profile_root*

The following list shows the default directory for a WebSphere Application Server Network Deployment profile named *profile_name*:

- AIX `/usr/IBM/WebSphere/AppServer/profiles/`*`profile_name`*
- Linux `/opt/IBM/WebSphere/AppServer/profiles/`*`profile_name`*

*installed_apps_root*

The following list shows the default directory for installed applications within a profile named *profile_name*:

- AIX `/usr/IBM/WebSphere/AppServer/profiles/`*`profile_name`*`/installedApps/`*`cell_name`*`/`
- Linux `/opt/IBM/WebSphere/AppServer/profiles/`*`profile_name`*`/installedApps/`*`cell_name`*`/`

*db_client_root*

The following list shows default installation root directories for the database clients:

- DB2 AIX `/usr/IBM/db2/V9.5`
- DB2 Linux `/opt/IBM/db2/V9.5`
- Oracle `/home/oracle/app/oracle/product/11.1.0`

## Glossary

This glossary contains terms that pertain specifically to the IBM WebSphere software for Telecom: IBM WebSphere IP Multimedia Subsystem Connector V6.2, IBM WebSphere Presence Server V7.0, IBM WebSphere Telecom Web Services Server V7.0, and IBM WebSphere XML Document Management Server V7.0.

The glossary also contains relevant terms from the IBM English Terminology Database.

### A

**Administrative console**

A graphical interface that guides the user through systems administration tasks such as deployment, configuration, monitoring, starting and stopping applications, services, and resources.

**Application Manager**

In Common Desktop Environment (CDE), a window containing objects representing the system actions available to you.

**application programming interface (API)**

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

### B

## C

**Call Notification**

A Parlay X Web service that notifies Web clients of specific call events established through the SIP protocol for a specific called party. Call Notification supports regular SIP and IMS call flows.

**CDMA2000**

A set of 3G standards based on earlier 2G CDMA technology.

**charge header support vector utility**

A utility class that handles Session Initiation Protocol (SIP) messages, for charging interactions.

**Charging Collection Function (CCF)**

Defined by the 3GPP group as the entity that receives information through Diameter messages pertaining to Charging Data Records.

**cluster**

A group of servers that are managed together and participate in workload management. See also horizontal cluster, vertical cluster.

**code division multiple access (CDMA)**

A form of multiplexing where the transmitter encodes the signal using a pseudo-random sequence, which the receiver also knows and can use to decode the received signal. Each different random sequence corresponds to a different communication channel.

**common base event**

A specification based on XML that defines a mechanism for managing events, such as logging, tracing, management, and business events, in business enterprise applications.

**common event infrastructure (CEI)**

A core technology of the IBM Autonomic Computing initiative that provides basic event management services, including consolidating and persisting raw events from multiple, heterogeneous sources and distributing those events to event consumers.

## D

**demilitarized zone (DMZ)**

A configuration including multiple firewalls to add layers of protection between a corporate intranet and a public network, like the Internet.

## E

**Enhanced Data Rate for GSM Evolution (EDGE)**

A development of GSM that allows for the faster delivery of advance mobile services such as full multimedia messaging.

**Enterprise JavaBeans**

A component architecture defined by Sun Microsystems for the development and deployment of object-oriented, distributed, enterprise-level applications.

**event state compositor (ESC)**

A server that processes PUBLISH requests and is responsible for composing an event state into a complete, composite event state of a resource.

## F

**frequency division duplex (FDD)**
The application of FDMA to separate outbound and returning signals. The uplink and downlink subbands are said to be separated by the ″frequency offset.″

**frequency division multiple access (FDMA)**
An access technology that is used by radio systems to share the radio spectrum. The terminology “multiple access” implies the sharing of the resource among users, and “frequency division” describes how the sharing is done by allocating users with different carrier frequencies of the radio spectrum.

## G

**General Packet Radio Service (GPRS)**
A mobile data service available to users of GSM mobile telephones. It is often described as ″2.5G.″ that is, a technology between the second (2G) and third (3G) generations of mobile telephony. It provides moderately fast data transfer by using unused TDMA channels in the GSM network.

**Global System for Mobile Communications (GSM)**
A second-generation (2G) standard for digital cellular telephone systems, which originated in Europe and is now used in countries across the globe. GSM networks use digital signals and narrowband TDMA, in conformance to a standard developed by the 3GPP, to support voice, data, text, and facsimile transmissions. The world’s most popular standard for mobile telephones, GSM service is used by more than 1.5 billion people across more than 210 countries and territories.

**Groupe Special Mobile (GSM)**
See Global System for Mobile Communications (GSM).

## H

**home subscriber server (HSS)**
The server that manages the database of all subscriber and service data in an IMS network. Parameters include user identity, allocated S-CSCF name, roaming profile, authentication parameters, and service information.

**horizontal cluster**
A cluster in which the cluster members exist on multiple physical servers, effectively and efficiently distributing the workload of a single instance. Horizontal clustering provides the ability to build in redundancy and failover, to easily add new members to increase capacity, and to improve scalability by adding heterogeneous systems into the cluster. See also vertical cluster.

**hypertext transfer protocol (HTTP)**
An Internet protocol that is used to transfer and display hypertext and XML documents on the Web. Hypertext Transfer Protocol Secure (HTTPS).

## I

**IMS Application Server (AS)**
Defined by the 3GPP to be the functional component that invokes applications (usually SIP applications) that provide services to IMS users.

**Institute of Electrical and Electronics Engineers (IEEE)**
A professional society accredited by the American National Standards Institute (ANSI) to issue standards for the electronics industry.

**Internet Engineering Task Force (IETF)**
The task force of the Internet Architecture Board (IAB) that is responsible for solving the short-term engineering needs of the Internet. The IETF consists of numerous working groups, each focused on a particular problem. Internet standards are typically developed or reviewed by individual working groups before they can become standards.

**IP Multimedia Subsystem (IMS)**
A network services architecture defined by 3GPP that enables support for IP multimedia applications based on SIP and IETF Internet protocols. IMS can use a variety of access methods, including wire-line IP, IEEE 802.11, 802.15, CDMA, and packet data transmission systems such as GSM, EDGE, and UMTS.

## J

**Java 2 Platform, Enterprise Edition (J2EE)**
An environment for developing and deploying enterprise applications, defined by Sun Microsystems Inc.

**Java API for XML-based RPC (JAX-RPC)**
A specification that describes application programming interfaces (APIs) and conventions for building Web services and Web service clients that use remote procedure calls (RPC) and XML. JAX-RPC is also known as JSR 101.

**Java authentication authorization service (JAAS)**
In J2EE technology, a standard API for performing security-based operations. Through JAAS, services can authenticate and authorize users while enabling the applications to remain independent from underlying technologies.

**Java Database Connectivity (JDBC)**
An industry standard for database-independent connectivity between the Java platform and a wide range of databases. The JDBC interface provides a call-level API for SQL-based and XQuery-based database access.

**Java Management Extensions (JMX)**
A means of doing management of and through Java technology. Developed by Sun Microsystems, Inc., and other leading companies in the management field, JMX is a universal, open extension of the Java programming language for management that can be deployed across all industries, wherever management is needed.

**Java Messaging Service (JMS)**
An application programming interface that provides Java language functions for handling messages.

**Java Naming and Directory Interface (JNDI)**
An extension to the Java platform that provides a standard interface for heterogeneous naming and directory services.

**Java virtual machine (JVM)**
A software implementation of a central processing unit that runs compiled Java code (applets and applications).

## K

## L

**Lightweight Directory Access Protocol (LDAP)**
An open protocol that uses TCP/IP to provide access to directories that support an X.500 model and that does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). For example, LDAP can be used to locate people, organizations, and other resources in an Internet or intranet directory.

**location generator**
The entity that initially determines or gathers the location of the target and creates location objects that describe the location of the target.

**location object**
An object that conveys location information (and possibly privacy rules) to which Geopriv security mechanisms and privacy rules are to be applied.

**location recipient**
The entity that receives location information. It might have asked for this location explicitly (by sending a query to a location server), or it might receive this location asynchronously.

**location server**
an element that receives publications of Location Objects from Location Generators and may receive subscriptions from Location Recipients. An entity that receives location objects published by a location generator, receives queries from location recipients, and applies privacy rules designed by the rule maker, typically the target to whose location information the rules apply.

## M

**mediation primitives**
Program components that can be assembled into customized message-processing flows in conjunction with the IBM WebSphere Telecom Web Services Server (TWSS) Access Gateway.

**message-driven bean (MDB)**
An enterprise bean that provides asynchronous message support and clearly separates message and business processing.

**mixed-media multilink transmission group (MMMLTG)**
A multilink transmission group that contains links of different medium types (for example, token-ring, switched SDLC, nonswitched SDLC, and frame-relay links).

**MLP** Mobile Location Protocol, an Open Mobile Alliance (OMA) specification.

## N

**natural language support (NLS)**
The ability for a user to communicate with hardware and software products in a language of choice to obtain results that are culturally acceptable.

## O

**Open Mobile Alliance (OMA)**
A standards body that develops open standards for the mobile phone industry.

## P

**Parlay**
A set of specifications for application programming interfaces (APIs) for managing network services such as call control, messaging, and content-based charging.

**Parlay Connector**
A Parlay Connector is the primary system component of Telecom Web Services Server (TWSS) that provides connectivity to a Parlay gateway by using a distributed communication protocol, most commonly Common Object Request Broker Architecture (CORBA).

**Parlay gateway**
A server that hosts the service implementations for the Parlay API. The TWSS Parlay Connector communicates with the Parlay gateway over CORBA. The Parlay API consists of various telecom service APIs which provide an abstract interface to network elements deployed in the service provider network. Some TWSS Web service implementations utilize the Parlay Connector to enable using the Parlay API to support the functions exposed as Parlay X Web services.

**Parlay X**
A set of Web services designed to enable software developers to use telecommunication capabilities in applications.

**Presence**
A Parlay X Web service that allows client applications to use Web services to subscribe to a presentity, synchronously query the current presence information for a presentity, receive asynchronous notifications about changes in the presence information for a presentity, and unsubscribe from a presentity.

**presence agent (PA)**
A SIP user agent that is capable of receiving SUBSCRIBE requests, responding to them, and generating notifications of changes in presence state. A presence agent must have knowledge of the presence state of a presentity. This means that it must have access to presence data manipulated by PUAs for the presentity.

**presence information**
Information comprising one or more presence tuples.

**presence server**
A service that accepts, stores, and distributes presence information.

**presence tuple**
A set of data comprising a status, an optional communication address, and optional other presence information.

**presence user agent (PUA)**
A SIP user agent that manipulates presence information for a presentity. This manipulation can be the side effect of some other action (such as sending a SIP REGISTER request to add a new Contact) or can be done explicitly through the publication of presence documents. A presentity can have one or more PUAs. This means that a user can have many devices

(such as a cell phone and personal digital assistant (PDA), each of which is independently generating a component of the overall presence information for a presentity. PUAs push data into the presence system but are outside it; they do not receive SUBSCRIBE messages or send NOTIFY messages.

**presentity**
A presence entity, a software entity that provides presence information to a presence service.

**public switched telephone network (PSTN)**
A communications common carrier network that provides voice and data communications services over switched lines.

## Q

## R

**registrar server**
An SIP server that keeps track of where a user can be contacted and provides that information to callers. A SIP phone must register its current location with a registrar server to allow calls to be made to it using a phone number or alias. Without a registrar server, the caller would need to know the correct IP address and port of the telephone.

**resource list server (RLS)**
A server that accepts subscriptions to resource lists and sends notifications to update subscribers of the state of the resources in a resource list.

## S

**Service Component Architecture (SCA)**
A set of specifications, published by the Open Service Oriented Architecture collaboration (OSOA), that describe a model for building applications and systems that builds on Service-Oriented Architecture (SOA) specifications.

**Service Data Object (SDO)**
An open standard for enabling applications to handle data from heterogeneous data sources in a uniform way. SDO incorporates J2EE patterns but simplifies the J2EE data programming model.

**Service Policy Manager**
A component of WebSphere Telecom Web Services Serverthat provides a storage capability and access mechanism to enable the definition of requesters, services, and subscriptions that associate services with requesters.

**service-oriented architecture (SOA)**
A conceptual description of the structure of a software system in terms of its components and the services they provide, without regard for the underlying implementation of these components, services and connections between components.

**serving-call session control function (S-CSCF)**
A server that acts as the central node of the signalling plane in a SIP network to register users and determine routing of messages. The S-CSCF also performs additional functions like providing routing services, enforcing policies, and providing billing information.

**servlet**
A Java program that runs on a Web server and extends the server's functionality by generating dynamic content in response to Web client requests. Servlets are commonly used to connect databases to the Web.

**Session Initiation Protocol (SIP)**
An Internet Engineering Task Force (IETF) standard protocol for initiating an interactive user session that involves multimedia elements such as video, voice, chat, gaming, and virtual reality.

**Short Message Peer-to-Peer Protocol (SMPP)**
A telecommunications industry protocol for exchanging Short Message Service (SMS) messages between SMS peer entities such as short message service centers.

**Short Message Service (SMS)**
A service that is used to transmit text to and from a mobile phone.

**Simple Object Access Protocol (SOAP)**
A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP can be used to query and return information and invoke services across the Internet.

**SIP Instant Messaging and Presence Leveraging Extensions (SIMPLE)**
An architecture for the implementation of a traditional buddylist-based instant messaging and presence application with SIP.

**stateless SIP proxy**
A proxy that receives SIP requests and forwards the request to a particular SIP container in a cluster, based on SIP dialog affinity, load balancing, and failover considerations.

## T

**target**
(1) The destination for an action or operation. (2) An entry point into Partner Gateway. It is an instance of a receiver configured for a particular deployment; each target supports documents sent using a single transport type and multiple targets can exist for a given transport type, one for each document format. See also receiver.

**Telecom Web Services Access Gateway**
Provides policy-driven traffic monitoring, message capture, authorization, and management capabilities. These services are provided at the application layer, and they are enforced for each Web service request using knowledge of the requester, target service, and invoked operation.

**WebSphere Telecom Web Services Server (TWSS)**
WebSphere Telecom Web Services Server provides a middleware infrastructure for managing Web service access and an environment for hosting Web service API implementations, which provides flexibility for construction of tailored message processing logic in accordance with service provider network policies.

**Terminal Location**
A component of WebSphere Telecom Web Services Server that enables applications to send Web services requesting the Terminal Location services defined by the Parlay X 2.1 specification, and to register for Terminal Location Notifications.

**Third Party Call**
A Parlay X Web service that provides the ability to initiate a call from a network entity between two different users or user agents

**time division multiple access (TDMA)**
A technology for shared-medium (usually radio) networks. It allows several users to share the same frequency by dividing it into different time slots. The users transmit in rapid succession, one after the other, each using their own timeslot. This lets multiple users share the same transmission medium (for example, radio frequency) while using only the part of its bandwidth they require. In radio systems, TDMA is almost always used alongside frequency division multiple access (FDMA) and frequency division duplex (FDD); the combination is referred to as FDMA/TDMA/FDD.

## U

**Universal Mobile Telecommunications System (UMTS)**
The third generation mobile telecommunications standard, defined by the ITU, that increases transmission speed to 2 Mbps per mobile user and establishes a global roaming standard.

**user agent client (UAC)**
In SIP, a client application that initiates the SIP request.

## V

**vertical cluster**
A cluster in which the cluster members exist on a single physical server. Vertical clustering can be an effective way to take full advantage of a multiprocessor server. See also horizontal cluster.

## W

**W-CDMA (wideband code division multiple access)**
A wideband spread-spectrum 3G mobile telecommunication air interface that uses CDMA. W-CDMA is the technology behind UMTS and is one of the interfaces used in cellular networks.

**Web Services Description Language (WSDL)**
An XML-based specification for describing networked services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

**WebSphere Integration Developer (WID)**
An integrated development and test environment and can be used as a visual editor when working with WebSphere Telecom Web Services Server mediation primitives to create customized flows.

**WebSphere software for Telecom (WsT)**
An IBM product suite that extends the industry leading WebSphere Application Server platform to deliver a fully IMS standards-compliant SIP application server, helping customers develop and deploy IP Multimedia Subsystem (IMS) compliant applications.

## X

**XCAP server**

An HTTP server that acts as a repository for collections of XML documents. It manipulates user data such as authorization policy, resource list, and other XML resources and provides access to these resources through the HTTP protocol.

**XML Configuration Access Protocol (XCAP)**

An IETF specification (RFC 4825) that allows a client to read, write, and modify application configuration data stored in XML format on a server.

**XML Document Management (XDM)**

An OMA specification for accessing and manipulating XML documents that are stored in repositories in a network. Using XDM, an application can work with individual XML elements and attributes instead of entire documents. The XDM specification is based on the IETF XML Configuration Access Protocol (XCAP).

## Y

## Z

## Numerics

**3rd Generation Partnership Project (3GPP)**

A collaboration agreement established in December 1998 through which ETSI (Europe), ARIB/TTC (Japan), CCSA (China), ATIS (North America), and TTA (South Korea) are making a globally applicable third-generation (3G) mobile phone system specification within the scope of the ITU's IMT-2000 project. 3GPP specifies the standards for UMTS.

**3rd Generation Partnership Project 2 (3GPP2)**

A collaboration agreement established in December 1998 through which ARIB/TTC (Japan), CCSA (China), TIA (North America), and TTA (South Korea) are making a globally applicable third-generation (3G) mobile phone system specification within the scope of the ITU's IMT-2000 project. 3GPP2 specifies the standards for CDMA2000.

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

400
AIX
DB2
Everyplace
IBM
pSeries
SP
Tivoli
WebSphere

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

# Readers' Comments — We'd Like to Hear from You

**IBM WebSphere Telecom Web Services Server**
**IBM WebSphere Telecom Web Services Server**
**Version 7.1**

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: 1-800-227-5088 (US and Canada)

If you would like a response from IBM, please fill in the following information:

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | E-mail address |

## Readers' Comments — We'd Like to Hear from You

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department 6R4A
P.O. Box 12195
Research Triangle Park, NC 27709-9990

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line