WebSphere® IBM WebSphere IP Multimedia Subsystem Connector

Version 6.2

IBM WebSphere IP Multimedia Subsystem Connector

Fourth Edition (April 2010)

This edition applies to IBM WebSphere IP Multimedia Subsystem Connector , Version 6.2, 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

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. IBM WebSphere IMS Service Control Interfaces Component

IBM® WebSphere® IMS™ Service Control Interfaces Component (also referred to as ISC Interfaces) is an integral part of WebSphere Application Server versions 6.1 and 7.0. It defines how an IMS CSCF interacts with service platforms, and it is licensed for use only through the IBM WebSphere IP Multimedia Subsystem Connector Version 6.2 license.

ISC Interfaces define how the WebSphere Application Server communicates with the Call Session Control Function (CSCF) in the IMS control plane. ISC Interfaces is not a callable application programming interface (API): rather it is a defined mechanism that specifies the way in which the CSCF and the IMS Application Server interact with each other.

The ISC interface is formally defined by 3GPP and 3GPP2 in the following standards:

> 3GPP: 3GPP TS 23.228 Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 6)
>
> 3GPP2: 3GPP2 X.S0013-002 All-IP Core Network Multimedia Domain (MMD); IP Multimedia Subsystem; Stage 2

ISC Interfaces is a bidirectional interface that uses SIP and is based upon IETF RFC 3261, which gives specifications for standardized SIP messages exchanged between the CSCF and IMS Application Server. The role of the ISC Interfaces is to support service invocation, present SIP parameters to applications, and interact with the service proxy for service provisioning.

Service Invocation and Interaction

The service platform triggers an initial SIP request to the Serving Call Session Control Function (S-CSCF). The CSCF proxies the service request to the corresponding application based on triggers. The IMS Application Server acts as a user agent, proxy server, and B2BUA (back-to-back user agent). The IMS Application Server may record and route SIP requests to stay in the signaling path, and the CSCF maintains the states between dialogs sent to or from applications, and it interacts with the service proxy for service provisioning.

Presentation of SIP Parameters

The ISC interface supports Service Point Triggers (SPT) for the SIP methods at the CSCF (for example: REGISTER, INVITE, SUBSCRIBE, MESSAGE). Data in the SPTs include:

- Presence or absence of any header
- Content of any header
- Direction of the request
- Session description information (SDP)

Service Proxy Function

In the IMS architecture, the S-CSCF provides a common protocol to SIP-based application services running on platforms such as WebSphere Application Server (and legacy services running on Intelligent Network (IN) and Open System Architecture (OSA) platforms). Filter criteria, which are based on SPTs and described in XML, are downloaded to the S-CSCF from the Home Subscriber Server (HSS), thereby defining which service platform or platforms are used; and in which order, based on information received by the S-CSCF. Initial or subsequent triggers may be applied in the IMS Application Server itself. The CSCF then forwards the request to the appropriate IMS Application Server, such as WebSphere Application Server. The IMS Application Server receives the request, applies the business logic for the application, and appropriately routes the request. The ISC interface governs these interactions between the IMS Application Server and the CSCF.

Based on the ISC interface, there are several ways in which the IMS Application Server might interact with the CSCF:

- Act as a terminating user agent (UA)
- Act as an originating user agent to originate traffic on behalf of a user
- Receive requests
- Serve as a proxy function
- Act as a third party call control applet

3GPP defines a list of Private Headers (P-Headers) to SIP, which allow for control mechanisms and others in an IMS environment. As SIP requests and messages are processed in the IMS control plane, these P-Headers are inserted and made available to the IMS Application Server. The IMS Application Server can then act on them, enhance them, and provide information using P-Headers.

The following P-Headers are the most relevant to the IMS Application Server:

- P-Asserted-Identity (RFC3325): carries valid and authenticated public user identity from the IMS control plane to the IMS Application Server.
- P-Charging-Vector (RFC3455): carries charging correlation information from IMS control plane to the IMS Application Server.
- P-Charging-Function-Addresses (RFC3455): carries offline and online charging function addresses from the IMS control plane to the IMS Application Server.

The following P-Headers are also visible to the IMS Application Server and the user environment.

- P-Access-Network-Info (RFC3455): carries information of the access network from the user environment to the IMS control plane and from the IMS control plane to the IMS Application Servers, which allows the user environment to provide information related to the access network it is using (such as cell ID).
- P-Called-Party-ID (RFC3455): carries the target public user identity from the IMS control plane to the user environment, which allows the terminating user environment to learn the dialed public user identity that triggered the call. This field may be seen at the IMS Application Server when the IMS Application Server is the called party (such as the destination of the session), but not in other scenarios (such as when the IMS Application Server is just a proxy in the chain of proxies in the path towards a user environment).

Chapter 2. Trust Association Interceptor security component

The Trust Association Interceptor security component is shipped with the IBM WebSphere IP Multimedia Subsystem Connector and is intended to enhance the overall authentication security for the IBM WebSphere software for Telecom.

Introduction to TAI

The Trust Association Interceptor provides a way for users to become known to WebSphere Application Server without having to re-authenticate. The result is a simpler, yet still secure, authentication process for requests flowing throughout the network.

Overview

When a user was authenticated by an authentication system other than WebSphere Application Server, it is possible to inform WebSphere Application Server of the user's identity information without requiring the user to re-authenticate. This is known as identity assertion.

The Trust Association Interceptor component intercepts HTTP and SIP service requests to all IMS service plane components, verifies the pre-hop sender, consumes identity information passed to it in HTTP/SIP headers, and propagates this information as WebSphere Application Server-rich security information.

Supporting trust association generally implies that WebSphere Application Server processes service requests in conjunction with a reverse proxy security server (RPSS). Typically, WebSphere Application Server and the proxy server engage in a contract in which the product gives its full trust to the proxy server and the proxy server applies its authentication policies on every Web request that is dispatched to WebSphere Application Server. This trust is validated by the interceptors that reside in the product environment for every request received. The method of validation is agreed upon by the proxy server and the interceptor.

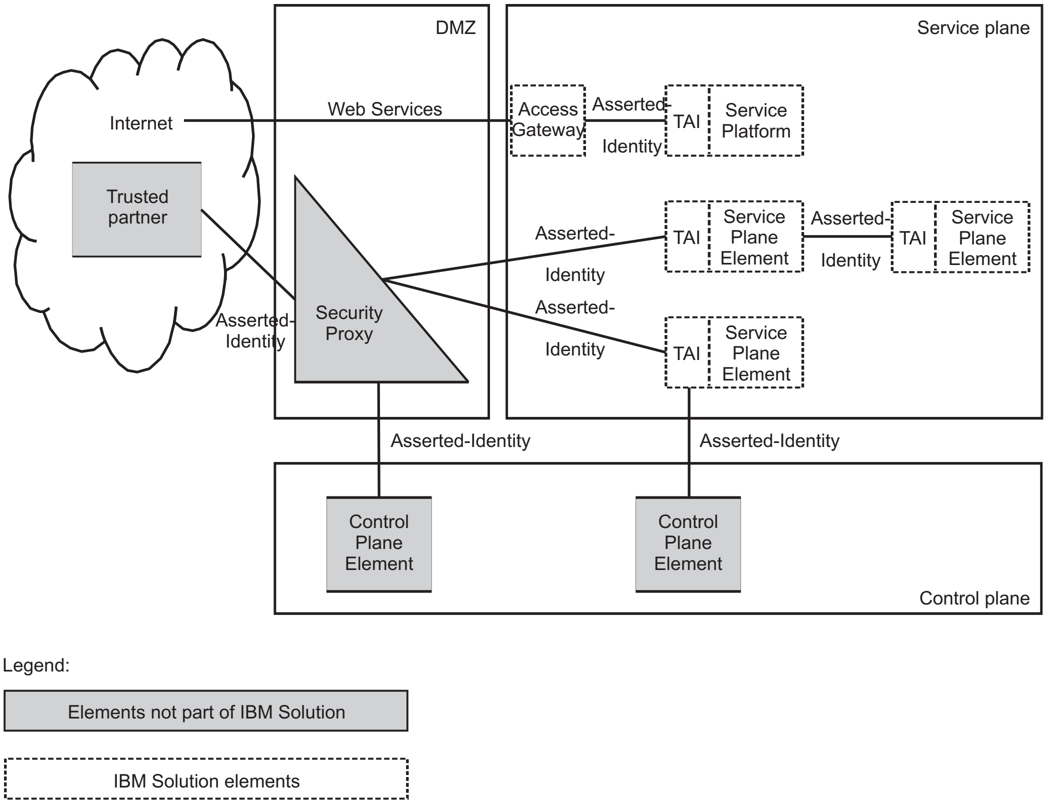
The Trust Association Interceptor implementation does not require the presence of an RPSS, but is designed with a security proxy in mind. The Trust Association Interceptor processes all authenticated requests originating in the trusted network that contain the expected HTTP/SIP asserted identity headers. It is assumed that a user is authenticated prior to the invocation of the interceptor by the security proxy or a control plane element. The interceptors do not perform authentication themselves.

The purpose of an interceptor is to validate requests based on HTTP/SIP headers and map user information to WebSphere Application Server-rich security information. Running in trust association mode does not prohibit WebSphere Application Server from accepting requests that did not pass through the proxy server. An interceptor is not needed for validating trust. However, it is possible to configure WebSphere Application Server to strictly require that all HTTP requests go through a reverse proxy server. In this case, all requests that do not come from a proxy server are immediately denied by WebSphere Application Server. In the IMS network, requests from the Internet to the service platform, coming through the Telecom Web Services Server (TWSS) Access Gateway, are allowed to bypass the security proxy, as are previously authenticated calls from the control plane.

System Structure

As this diagram shows, the solution includes a DMZ (dual firewall) to isolate protected service plane elements in the trusted domain from potential threats. A security proxy (RPSS) is implemented as a border element within the DMZ that authenticates users and propagates user information for all requests that pass through. This security proxy is not formally specified nor required by the IBM solution, except that its interface requirements must correspond with the Trust Association Interceptor implementation.

The figure also shows that a common, custom Trust Association Interceptor exists within WebSphere Application Server to intercept HTTP and SIP traffic in the IMS-trusted domain.



This implementation requires all IMS elements (base WebSphere Application Server, the control plane, security proxy, and the invoked service plane application) to propagate asserted identity tokens on every service request to any other service plane component within the trusted domain. This dependency is important to note because it requires coordination among all of the IBM WebSphere software for Telecom products.

X-3GPP-Asserted-Identity and P-Asserted-Identity headers are supported, by default, for HTTP and SIP requests, respectively. (Refer to the 3GPP implementation for IETF RFC 3325.) However, the type of asserted identity header is configurable for both HTTP and SIP requests to support components with different standard asserted identity headers. For example, the Aggregation Proxy component passes an X-XCAP-Asserted-Identity header to the IBM XDMS application.

The IMS control plane inserts a P-Asserted-Identity header for authenticated subscribers and may or may not go through the security proxy. All device-originating traffic is routed through the control plane (S-CSCF) before reaching the IBM service plane.

For Web services access on the service platform, the TWSS Access Gateway is used for front-end authentication and the DMZ/security proxy is circumvented.

Configuring the Trust Association Interceptor

The Trust Association Interceptor (TAI) contains two interceptors to process the incoming requests: HttpInterceptor and SipInterceptor. Each interceptor has properties associated with it. Use the WebSphere Integrated Solutions Console to set the properties.

The interceptor properties are created and configured from the Integrated Solutions Console during the installation process by entering the interceptor Name, Value, and Description.

Note:

- The allowedSenderList property is required at a minimum.
- For information about the installation process, refer to the Installing section for each component.

This topic provides the description of the properties and their default values.

HTTP Properties

The HttpInterceptor for the Trust Association Interceptor security component contains the following properties:

Table 1. HttpInterceptor properties

Parameter	Type	Description
enableSenderVerification	Boolean	Default: true. When true, enables the verification of the identity of the pre-hop sender of the incoming request. The user may want to disable this verification when transport-level security or IPSec is enabled on the service platform.

Table 1. HttpInterceptor properties (continued)

Parameter	Type	Description
allowedSenderList	string	Default: null. Comma-delineated list of hosts that the interceptor considers trusted. When the enableSenderVerification property is false, this property is ignored. Acceptable input is hostname or IP address. Wildcards and masking are allowed. The following is a list of acceptable examples: Hostnames: *@us.example.com, *.example.com, dyn94158159.example.com IPv4 addresses: 9.41.x.x, 9.41.57.154, 9.41.57.154/20, 9.41.57.154/255.255.240.0 IPv6 addresses: 2002:092A:8F7A:0000:0000:0020:0000:0001, 2002:92A:8F7A:0:0:20:0:1 2002:92A:8F7A:0:0:20::1, 2002:92A:8F7A::20:0:1 (the four addresses are different representations of the same address) 2002:92A:8F7A::20:0:1/60 The following indicates that ALL prehop senders will be allowed: *, x.x.x.x
assertedIdentityHeaderType	string	Default: X-XCAP-Asserted-Identity, X-3GPP-Asserted-Identity. Comma-delineated list of allowed types of asserted identity headers supported in this interceptor implementation. For example, a different X-XCAP-Asserted-Identity header may be required in the IBM XDMS implementation. Multiple values are allowed to accommodate ambiguous specification guidelines. A maximum of 2 entries is allowed.
enableDefaultRoleMapping	Boolean	Default: true. Enables default role mapping, which maps all users to the All authenticated group. Enabling this property prevents WebSphere Application Server from invoking the user registry to create a Subject. Applications that depend on WebSphere Application Server roles may want to disable this property.
enableMultipleIDMapping	Boolean	Default: true. Enables stripping the protocol scheme from the SIP URI, to obtain an ID that may match other IDs belonging to the same user. This applies to situations where there are multiple public IDs for a single user.

Table 1. HttpInterceptor properties (continued)

Parameter	Type	Description
handleUnauthenticatedUser	Boolean	Default: false. Enables processing of messages from unauthenticated users. The unauthenticated user will be mapped to the Principal with the value defined in the unauthenticatedPrincipal property.
anonymousUserID	string	Default: anonymous. The user ID that identifies an anonymous user.
anonymousPrincipal	string	Default: anonymous.invalid. The value that is mapped to the WebSphere Application Server Principal for an anonymous user.
unauthenticatedPrincipal	string	Default: anonymous.invalid. The value that is mapped to the WebSphere Application Server Principal for an unauthenticated user. The default is to handle unauthenticated users the same as anonymous users.
allowedAssertedProxyUsers	string	Default: null. Comma-delineated list of allowed asserted identity values that indicate the request originates with a proxy, not an end user. Accepted format is hostname, email name, or URI. Wildcards are acceptable (for example: *@us.example.com, *.example.com). A maximum of 30 entries is allowed.
enableLTPABypass	Boolean	Default: False. When set to true, the TAI includes a WSCredential in the TAIResult's Subject. This prevents a JAAS login and a look-up in a user registry downstream by WebSphere security. It further prevents WAS from adding an LTPA token to the message header. Avoiding expensive LTPA calls by WebSphere, might improve IMS systems performance. When this is set to true, you must also disable **security attribute propagation** from the Integrated Solutions Console: 1. Click **Security** → **Secure Administration** → **web security** → **single sign-on (SSO)**, and clear the **Web inbound security attribute propagation** check box. 2. Click **Security** → **Secure Administration** → **RMI/IOP security** → **CSIv2 inbound authentication**, and clear the **security attribute propagation** check box. 3. Click **Security** → **Secure Administration** → **RMI/IOP security** → **CSIv2 outbound authentication**, and clear the **security attribute propagation** check box.

SIP Properties

The SipInterceptor for the Trust Association Interceptor security component contains the following properties:

Table 2. SipInterceptor properties

Parameter	Type	Description
enableSenderVerification	Boolean	Default: true. When true, enables the verification of the identity of the pre-hop sender of the incoming request. The user may want to disable this verification when transport-level security or IPSec is enabled on the service platform.
allowedSenderList	string	Default: null. Comma-delineated list of hosts that the interceptor considers trusted. When the enableSenderVerification property is false, this property is ignored. Acceptable input is hostname or IP address. Wildcards and masking are allowed. The following is a list of acceptable examples: Hostnames: *@us.example.com, *.example.com, dyn94158159.example.com IPv4 addresses: 9.41.x.x, 9.41.57.154, 9.41.57.154/20, 9.41.57.154/255.255.240.0 IPv6 addresses: 2002:092A:8F7A:0000:0000:0020:0000:0001, 2002:92A:8F7A:0:0:20:0:1 2002:92A:8F7A:0:0:20::1, 2002:92A:8F7A::20:0:1 (the four addresses are different representations of the same address) 2002:92A:8F7A::20:0:1/60 The following indicates that ALL prehop senders will be allowed: *, x.x.x.x
assertedIdentityHeaderType	string	Default: P-Asserted-Identity. Comma-delineated list of allowed types of asserted identity headers supported in this interceptor implementation. For example, a different X-XCAP-Asserted-Identity header may be required in the IBM XDMS implementation. Multiple values are allowed to accommodate ambiguous specification guidelines. A maximum of 2 entries is allowed.

Table 2. SipInterceptor properties (continued)

Parameter	Type	Description
enableDefaultRoleMapping	Boolean	Default: true. Enables default role mapping, which maps all users to the All authenticated group. Enabling this property prevents WebSphere Application Server from invoking the user registry to create a Subject. Applications that depend on WebSphere Application Server roles may want to disable this property.
pAssertedIdentityURIType	string	Default: sip. When two P-Asserted-Identity headers are present in the message, the user has the option of choosing which SIP URI type is used for the asserted identity. Possible values: sip, sips, and tel.
enableMultipleIDMapping	Boolean	Default: true. Enables stripping the protocol scheme from the SIP URI, to obtain an ID that may match other IDs belonging to the same user. This applies to situations where there are multiple public IDs for a single user.
handleUnauthenticatedUser	Boolean	Default: false. Enables processing of messages from unauthenticated users. The unauthenticated user will be mapped to the Principal with the value defined in the unauthenticatedPrincipal property.
anonymousUserID	string	Default: anonymous. The user ID that identifies an anonymous user.
anonymousPrincipal	string	Default: anonymous.invalid. The value that is mapped to the WebSphere Application Server Principal for an anonymous user.
unauthenticatedPrincipal	string	Default: anonymous.invalid. The value that is mapped to the WebSphere Application Server Principal for an unauthenticated user. The default is to handle unauthenticated users the same as anonymous users.
allowedAssertedProxyUsers	string	Default: null. Comma-delineated list of allowed asserted identity values that indicate the request originates with a proxy, not an end user. Accepted format is hostname, email name, or URI. Wildcards are acceptable (for example: *@us.example.com, *.example.com). A maximum of 30 entries is allowed.

Table 2. SipInterceptor properties (continued)

Parameter	Type	Description
enableErrorMessageReception	Boolean	Default: false. Enables processing of error messages that are sent from the SIP container to itself. An example is where a SIP message is sent to a SIP destination that is no longer attached. When the message times out, the SIP container sends an error message to itself. These messages lack p-asserted-identity headers and, as a result, are not normally processed by the TAI. If true, enables the processing of error messages that lack p-asserted-identity headers and are generated by the localhost. If false, any error messages sent by the SIP container to itself are processed using the same rules as all other messages. Use errorMessageStatusCodeList to designate which SIP error messages are to be processed.
errorMessageStatusCodeList	string	Default: 4xx. Comma-delineated list of message codes within the range of 400-499 (Client Error) and 500-599 (Server Error) to be processed when enableErrorMessageReception is true. You can use a wild card of ″*″, ″x″, or ″X″ to indicate a substitution for a single digit. For example, 4** would represent all 4xx error status Codes (400-499). A value of *** would represent all 4xx and 5xx codes. A maximum of 20 entries is allowed.
enableLTPABypass	Boolean	Default: False. When set to true, the TAI includes a WSCredential in the TAIResult's Subject. This prevents a JAAS login and a look-up in a user registry downstream by WebSphere security. It further prevents WAS from adding an LTPA token to the message header. Avoiding expensive LTPA calls by WebSphere, might improve IMS systems performance. When this is set to true, you must also disable **security attribute propagation** from the Integrated Solutions Console. For details, refer to the section titled *Disabling security attribute propagation*, later in this topic.

Disabling security attribute propagation

Follow these steps to disable security attribute propagation. This is required when **enableLTPABypass** is set to true.

If you are using WebSphere Application Server version 7.0.0.1:

1. Click **Security** → **Global Security** → **Web and SIP security** → **single sign-on (SSO)**, and clear the **Web inbound security attribute propagation** check box.
2. Click **Security** → **Global Security** → **RMI/IOP security** → **CSIv2 inbound communications**, and clear the **propagate security attributes** check box.
3. Click **Security** → **Global Security** → **RMI/IOP security** → **CSIv2 outbound communications**, and clear the **propagate security attributes** check box.

If you are using WebSphere Application Server version 6.1.0.21:

1. Click **Security** → **Secure Administration** → **Web security** → **single sign-on (SSO)**, and clear the **Web inbound security attribute propagation** check box.
2. Click **Security** → **Secure Administration** → **RMI/IOP security** → **CSIv2 inbound authentication**, and clear the **security attribute propagation** check box.
3. Click **Security** → **Secure Administration** → **RMI/IOP security** → **CSIv2 outbound authentication**, and clear the **security attribute propagation** check box.

Chapter 3. Introduction to the IBM WebSphere Diameter Enabler Component

The Diameter Enabler is an integral part of an IP multimedia subsystem (IMS) network solution in situations where applications need to retrieve and update network subscriber data.

Using the Diameter Enabler, you can build applications that:

- Send and receive subscriber profile information
- Send accounting information from your Home Subscriber Server (HSS)
- Send accounting information to your Charging Collection Function (CCF)
- Send subscriber account information between the Charging Trigger Function (CTF) and the Online Charging System (OCS)

The Diameter Enabler includes the following subcomponents:

Diameter Enabler base
Rf accounting Web service
Ro online charging Web service
Sh subscriber profile Web service

The Diameter Enabler base

The Diameter Enabler base interacts with one or more of the following entities, which are referred to as *Diameter peers* or simply *peers*: Home Subscriber Server (HSS), Charging Collection Function (CCF), Online Charging System (OCS), and Diameter agents.

The Diameter Enabler base establishes direct TCP connections with these entities and acts as a Diameter gateway: receiving and replying to Web service requests on the one side while sending and receiving Diameter packets on the other. This eliminates the need for you to develop applications that use the Diameter protocol.

Diameter Enabler supports inbound and outbound connections initiated by peers and external applications:

- Inbound: a connection is initiated by a remote peer to the Diameter Enabler base.
- Outbound: a connection is initiated by the Diameter Enabler base to a Diameter peer.

The connection is classified either as an inbound or outbound connection based on the side that initiates it. When the connection has been established, packets can be sent and received over that connection.

Diameter Enabler Web services

The Diameter Enabler provides application programming interfaces (APIs) that use Web services to help developers rapidly develop and deploy applications to access data from the Home Subscriber Server (HSS) and update data for the Charging Collection Function (CCF) and Online Charging System (OCS).

The Web Services Description Language (WSDL) is included for Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service. Each WSDL file describes the operations, parameters, and data types that comprise the interfaces of these Web services that can be executed by other applications.

The Rf accounting Web service serves as the interface for offline charging by implementing support for offline session charging and event charging. Rf provides the IMS Application Server application a Diameter messaging interface to enable sending accounting messages to the accounting or billing servers. The Rf accounting Web service receives Web service requests from the IMS Application Server, builds the appropriate Diameter message, sends the message to the Diameter Enabler base in the form of an accounting request, receives accounting answers from the Diameter Enabler base, and sends the accounting answers back to the IMS Application Server.

The Ro online charging Web service serves as the interface for online session and event charging. Ro provides the IMS Application Server application with a Diameter messaging interface to enable sending credit control authorization messages to the online charging servers. The Ro online charging Web service receives Web service requests from the IMS Application Server, builds the appropriate Diameter message, sends the message to the OCS, receives authorization answers from the OCS, and sends the results back to the IMS Application Server.

The Sh subscriber profile Web service serves as the interface between an IMS Application Server and the HSS where subscriber data is centrally stored. It provides retrieval and update APIs so that IMS Application Servers can upload data to and download data from the HSS; and it provides a subscription and notification service, so that the IMS Application Server can subscribe to data changes stored in the HSS.

The Sh subscriber profile Web service constructs Diameter messages on behalf of the IMS Application Server and sends them to the HSS by using the Diameter protocol. The Sh subscriber profile Web service sends the results of these Diameter message exchanges back to the IMS Application Server that initially called the Sh subscriber profile Web service. The IMS Application Server may or may not be located on the same machine as the Sh Web service.

The Diameter Enabler includes the following WSDLs:
- DiameterRfService.wsdl
- DiameterRoService.wsdl
- DiameterRoNotifyService.wsdl
- DiameterShService.wsdl
- DiameterShNotifyService.wsdl

For detailed information on WSDLs, refer to the topic *WSDL architecture* in the WebSphere Application Server Network Deployment information center.

Introduction to the Rf accounting Web service

IMS Application Server applications use the Rf accounting Web service to send accounting information to a Charging Collection Function (CCF). The Rf accounting Web service implements the operations required to perform offline accounting transactions. The operations allow an application to send event and session accounting records to the CCF.

The Rf accounting Web service interface is used only for offline charging. It includes the ability to send Accounting-Start, Accounting-Stop, and Accounting-Interim messages to an accounting server. It also has the ability to send an Accounting-Event notification to the accounting server. Offline charging can be used to track a single event, such as purchasing a cell phone ring tone, or to track a service that is used over time such as the start and end of a conference call. The *DiameterImsRfApi* defines the offline accounting transactions available through Web services.

Introduction to the Ro online charging Web service

IMS Application Server applications use the Ro online charging Web service to send authorization information to an Online Charging System (OCS). The Ro online charging Web service implements the operations required to perform online charging transactions. The operations allow an application to send session and event credit control charging information to the OCS. These operations are used to deplete quota from a subscriber’s account while the service is being used.

The Ro online charging Web service interface is used for online charging and includes the ability to send Credit Control-Initial, Credit Control-Update, and Credit Control-Termination messages to an online charging server. It also has the ability to send a Credit Control-Event notification to the online charging server. Online charging can be used to interactively charge a subscriber while a service is being provided. For example, a subscriber may have purchased 200 minutes for access to an online gaming service. The service depletes the subscriber’s quota as they use the service, and discontinues the service when the subscriber has used all 200 minutes. The *DiameterImsRoApi* defines the online charging transactions available through Web services.

In addition to the online event and session charging operations, the Ro online charging Web service interface also includes subscription and notification operations with which an IMS Application Server can subscribe to be notified when the CTF must reauthenticate with the OCS or when a reauthorization subscription expires.

Introduction to the Sh subscriber profile Web service

The Sh subscriber profile Web service provides IMS Application Servers secure access to subscriber data centrally stored on the Home Subscriber Server (HSS). This includes a subscription and notification service, so that the IMS Application Server instances can subscribe to data changes that occur in the HSS.

Although the Diameter Cx interface is used during authentication–between a Call Session Control Function (CSCF) and the Home Subscriber Server (HSS)–the IBM WebSphere Diameter Enabler Component does not implement this capability. Instead, the Diameter Enabler is used by an Application Server (AS) application to retrieve and update information about subscribers stored in a Home Subscriber Server (HSS).

This interface gives the IMS Application Server the ability to query location information, presence information, and subscriber entitled services.

The Sh subscriber profile Web service is an interface between the IMS Application Server and the HSS. The interface allows for four transaction types with a request and response transaction for each. These transactions provide the following functions:

- Sh-Pull (retrieving the profile of a subscriber) allows the WebSphere Application Server to query transparent and non-transparent data pertaining to a specific Subscriber or user. The WebSphere Application Server queries the HSS for this information.
- Sh-Update (updating the profile of a subscriber) allows the WebSphere Application Server to update the transparent information retained on the HSS for a specific subscriber or user.
- Sh-Subs-Notif (subscribing for notifications when a subscriber's profile changes) allows the IMS Application Server to register for notifications when particular transparent and/or non-transparent information is updated. There is an unsubscribe capability to end a particular subscription for notifications.
- Sh-Notif (The Notification initiated from the HSS when the subscriber's profile changes) is the transaction used by the HSS to inform the WebSphere Application Server that the information (transparent and non-transparent) specified by an earlier Sh-Subs-Notif request has been updated.

Connections, routes, channels, and call flows

The Diameter Enabler base utilizes connections as links between a Diameter Enabler base and a peer, as routes to determine the connection on which a Diameter packet should be sent, as channels to convert byte streams to and from Diameter packets, and as call flows to transmit and receive these packets.

A connection is the link between a Diameter Enabler base and a peer. The peer may be a Diameter server or a Diameter agent that is a hop in the path to the Diameter server. For each configured connection, there is exactly one TCP connection between the Diameter Enabler base and the peer that it communicates with.

Connections remain as long as both the Diameter Enabler base and the peer are operating. If neither party sends a packet over the connection for a specified period of time, a watchdog message is sent to ensure that both sides are still operating.

The Diameter layer uses routes to determine which connection it should send a packet on. To send Diameter application packets to a specific realm, you must configure at least one connection and one route that uses that connection.

Note: If the specified connection's destination is a Diameter agent, rather than a Diameter server, the Diameter packet passes through this connection as a single hop on its way to the Diameter server.

Each connection represents a chain of channels. These channel chains operate within the Channel Framework Architecture (CFA) and are comprised of Diameter Channels, SSL/TLS channels (optional), and TCP channels.

At the top of the channel chains, there are three applications: Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service. All three of these applications export APIs to external applications through Web services interfaces.

- The Diameter Enabler base manages connections, routes, and state machines to insure compliance with the Diameter protocol.
- Diameter Web services use the Diameter Enabler base to handle packets as they pass through the CFA.

- The Diameter channel is a serialization and de-serialization layer that converts inbound byte streams into Diameter packets and outbound Diameter packets into byte streams. No state is kept within the Diameter channel.

Diameter applications use call flows to transmit and receive packets for accounting, online charging, subscriber profile retrieval and updates, or subscribe and notify messages for subscriber profiles.

- For accounting messages, the Diameter Enabler base passes information through the Rf accounting Web service when sending messages to the CCF. The CCF receives the messages and sends its response back to the Rf accounting Web service, which responds to the client that initiated the Web service request.
- For online service requests, the Ro online charging Web service interface receives the Web service request from the client. The request is then processed and forwarded through the Diameter Enabler base to the appropriate OCS. The OCS receives the message and sends its response to the Ro online charging Web service, which responds to the client that initiated the Web service request.
- For subscribe/notify messages:
  - The Sh subscriber profile Web service interface receives subscription requests from the IMS Application Server. The requests are then forwarded as subscriptions to the HSS using the Diameter protocol. When an update occurs at the HSS, the HSS sends a notification to the Diameter Enabler informing it of the update. The Sh subscriber profile Web service then forwards the update to all IMS Application Servers that have subscribed to the update.
  - The Ro online charging Web service interface provides subscription and notification messages to the IMS Application Server. Notifications are sent when the CTF must reauthenticate with the OCS or when a reauthorization subscription expires.

Implementation in an existing network

The Diameter Enabler integrates with your existing network to enable access to a Home Subscriber Server (HSS), provide accounting data to a Charging Collection Function (CCF), and provide online charging data to an Online Charging System (OCS).

When the Diameter Enabler integrates with your existing network to enable access to HSSs, CCFs, and OCSs, you can configure the Diameter Web services to control the Diameter network to which they attach.

You can specify different Diameter networks (connections and routes) for the Rf accounting Web service, the Ro online charging Web service, and the Sh subscriber profile Web service. The network configurations are stored in files called `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties`, respectively.

The properties files are located in the following directory locations:

AIX Linux *was_profile_root*`/properties`

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*

Chapter 4. Planning for IBM WebSphere Diameter Enabler Component

For planning and installation, you must understand the hardware and software requirements, supported servers, installation topologies, security considerations, and configuration considerations.

Diameter Enabler includes Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service. If you are using the Ro or Sh Web services, you might have additional considerations identified in the planning and installation information.

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" on page 20

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

AIX®

Processor
Power 4 or Power 5

Physical memory
4 GB minimum, 2 GB per JVM recommended

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

Linux® on PowerPC®

Processor
Power 4 or Power 5

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)

4 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)

4 GB of free space recommended

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

Hyper-threading should be enabled

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.

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

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

“Operating systems”
“Application servers” on page 21
“Java version” on page 21
“Databases” on page 21

Operating systems

The following operating systems are supported:

Linux Red Hat Enterprise Linux AS 5.0 Update 2
Linux SUSE Linux Enterprise Server 10 SP 1
AIX AIX 5L 5.3 TL 07 SP 4

Application servers

One of the following application server offerings is required:

- WebSphere Application Server Network Deployment, version 7.0.0.1

  Note: WebSphere IMS Connector interim fix 1 is required if you plan to deploy on this version of WebSphere Application Server Network Deployment.
- 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.

Java™ version

The following IBM SDK 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.

Databases

Diameter Enabler requires a database only if you are using the subscribe and notify functions of Sh subscriber profile Web service or using Ro online charging Web service reauthorization.

The following databases are supported:

- DB2 IBM DB2® Enterprise Server Edition, version 9.5 FixPak 1
- Oracle Oracle Database, version 10.2.0.4 or 11.1.0.7

Planning your software topology

Before you start the installation process, it is important to understand the software topologies.

Diameter Enabler is deployed on the WebSphere Application Server. Before beginning the installation you must have WebSphere Application Server Network Deployment version 7.0.0.1 or 6.1.0.21 installed on the server.

Web services

Diameter Enabler provides the following Web services:

- Rf accounting Web service
- Ro online charging Web service
- Sh subscriber profile Web service

You can put the Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service in the same application server in a WebSphere Application Server installation. While the Rf, Ro, and Sh Web services reside on the same application server, each service will have a unique routing table.

Based on their configuration, the Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service can access entirely different Diameter server networks. They can also access the same Diameter server networks if those Diameter servers support Rf, Ro, and Sh interfaces.

RFC 3588 defines a maximum of one connection between any two Diameter peers. Therefore, if Rf, Ro, and Sh are used to access a peer, they must share a single connection. For shared connections, the connection configuration information must be identical in the `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` files.

The Rf, Ro, and Sh Web services are configured to define the realms and connections to the Diameter network. You can specify different realms and connections for each of the Web services.

Restrictions

The IMS Application Server application is typically not installed on the same application server as the Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service. You must install the Rf, Ro, and Sh Web services on the same server as Diameter Enabler base.

An IMS Application Server can be installed in a different distributed environment. The Diameter interfaces are Web services and can be called remotely from any server.

Do not install Diameter Enabler on both a cluster and a single server instance that are running in the same physical computer.

Database requirements

Database support is not needed for the Rf accounting Web service and for most of the Sh and Ro Web services. However, a database is required to use the subscribe and notify capabilities of the Sh subscriber profile Web service. The database provides reliable retention of the active subscriptions and the endpoints to contact when the Home Subscriber Server (HSS) issues a notification that subscriber data has changed.

A database is also required when you use the Ro online charging Web service reauthorization request (RAR) notification. This is an unsolicited message sent from the online charging server to the Charging Trigger Function. If you do not require the subscribe and notify functions of the Sh and Ro Web services, no database is required.

If you require a database, the database should be remote from the server where the Diameter Enabler base and the Sh and/or Ro Web services reside before you create a clustered server.

If you require a database, you must install the database clients or provide the JDBC JAR files on all servers where Diameter Enabler is installed.

Agents and servers

Diameter Enabler can interact directly with the Home Subscriber Server (HSS), the Online Charging System (OCS), or the Charging Collection Function (CCF) server. Alternatively, it can interact with Diameter agents–including proxies and relays. When there is a direct TCP connection between the Diameter Enabler and a Diameter server or Diameter agent, the server or agent is referred to as a *Diameter peer* or simply a *peer*.

Diameter Enabler can have exactly one connection to each Diameter peer.

The Rf accounting Web service provides an interface to the CCF, the Ro online charging Web service provides an interface to the OCS, and the Sh subscriber profile Web service provides an interface to the HSS. In most cases the CCF, OCS, and HSS will be on different physical servers.

Diameter Enabler converts a Web service request into one or more Diameter messages that it sends to the CCF, OCS, or HSS. Diameter Enabler also receives Diameter messages from the CCF, OCS, or HSS which it processes and returns to the applications on the IMS Application Server. The messages are returned through the Diameter Enabler Web services.

Routes

When Diameter Enabler sends a packet, it consults a routing table. The routing table is a simple structure based on the realm and Application Id that determines which connection to use when sending the packet. The route ties the final destination realm to the next hop connection for a given packet. Without a route in the routing table, the packet will not be forwarded even if a direct connection to the Diameter server is established.

Only one primary route to a destination realm can be configured within an environment (Rf, Ro, or Sh). A primary route is used to transmit Diameter packets whenever its underlying connection is active. Secondary routes are not required, but are recommended. A secondary route is used to transmit Diameter packets only when the primary route does not have an active or open connection. You can configure one or more secondary routes.

When the underlying connection of a primary route fails, the routing table automatically fails over to a secondary route where there is an active connection. If that connection fails and additional secondary routes are configured, the next secondary route is used until no more routes remain. If no primary or secondary routes are available, then a default route is used if one is defined. If no routes are available to carry the packet, an exception is sent to the user. If the primary route recovers from the failure, then all of the traffic is again routed through the primary route.

If no specific route is configured, you can configure a default route to forward packets. The algorithm selects a specific route first. If it is unable to find a match for the destination realm, it uses a default route. The default route can also be configured as a primary or secondary route.

The Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service have independent routing tables. Although all of these Web services use the Diameter Enabler base, the configuration and usage of the routing tables remain independent. Because of this, each Web service application can be started and stopped independently.

Realms

You must configure at least one realm, which can be either a specific realm or a default realm. Each Web service can support up to 10 realms. According to RFC 3588, you should have a primary and secondary route for each realm. The Diameter Enabler supports 10 realms for the Rf accounting Web service, 10 for the Ro online charging Web service, and 10 for the Sh subscriber profile Web service.

Therefore, you can configure 10 primary and 10 secondary routes for a total of 20 routes per Web service. If you are using all of the Web services (Rf, Ro, and Sh), then you can have a maximum of 60 routes total.

Supported network elements (CCF, OCS, and HSS)

Diameter Enabler functions as a Diameter client to the Charging Collection Function (CCF), Online Charging System (OCS), and Home Subscriber Server (HSS). The CCF, OCS, and HSS act as Diameter servers.

Diameter Enabler supports servers that adhere to the following specifications:

Charging Collection Function servers

- RFC 3588
- ETSI TS 132.260 V6.7.0 (2006-09)
- ETSI TS 132.299 V6.9.0 (2006-12)

Online Charging System servers

- RFC 3588
- RFC 4006
- ETSI TS 132.260 V6.7.0 (2006-09)
- ETSI TS 132.299 V6.9.0 (2006-12)

Home Subscriber Server

- RFC 3588
- ETSI TS 129.328 V6.12.0 (2006-12)
- ETSI TS 129.329 V6.7.0 (2006-09)

Evaluating your hardware environment

Diameter Enabler installs and runs as an application on WebSphere Application Server. It can be deployed on various hardware configurations.

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.

Single-server topology

The components are installed on the same server. You can use this type of installation to evaluate your system for a proof-of-concept, simple interoperability testing, or simple functionality testing of the solution.

In conjunction with the single-server topology, an IBM WebSphere Telecom Toolkit development environment can help you rapidly develop and deploy IMS Application Server applications. This toolkit is available as a free download. It is designed to reduce the time to develop applications that use the Diameter Enabler and other IBM WebSphere software for

Telecom. The toolkit includes sample applications that demonstrate how to invoke the Diameter Enabler Web services.

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.

The database is shared and clustered.

Development topology

An IBM WebSphere Telecom Toolkit development environment can help you rapidly develop and deploy IMS Application Server applications. This toolkit is available as a free download. It is designed to reduce the time to develop applications that use the Diameter Enabler and other IBM WebSphere software for Telecom program products. The toolkit includes sample applications that demonstrate how to invoke the Diameter Enabler Web services.

Scaling and reliability

Diameter Enabler can be configured to run in horizontal and vertical clusters.

To achieve scalability and reliability, Diameter Enabler utilizes the clustering ability provided with the WebSphere Application Server Network Deployment product. When you use databases, they should be configured for failover in order to increase reliability.

Other WebSphere Application Server methods, such as an IP sprayer or edge server, and other resource configurations, such as database or network topologies, can also be employed.

Scaling may be vertical only, horizontal only, or a combination. For both horizontal and vertical clusters, each server must have a unique OriginHostName and a unique HostIpAddress and port number pair.

Horizontal cluster

Horizontal clusters are servers on separate physical servers. Each server must have a unique IP address. Each server may be configured to listen on the same port number. 3868 is the recommended port number because the RFC references that port for Diameter connections.

Vertical cluster

Vertical clusters are application servers on the same physical server. Each application server may have the same IP address. The IP address and TCP port combination must be unique for every cluster member. If the IP address is the same for two or more vertical cluster members, the port must be different.

You can install the components into an existing cell environment where cell security is already configured. Alternatively, you can install the components in a local environment and then add the node to a cell later on.

Security considerations

Diameter Enabler uses the security provided by WebSphere Application Server Network Deployment. You should consider securing access to the various Diameter Enabler Web services, J2EE access to the internal Diameter Enabler infrastructure, security on connections, and end-to-end security for messages.

Application level security

Each Web service application has the ability to enable or disable application security. Because Diameter Enabler is an AAA protocol, it is recommended that these Web services be secured. You should enable WebSphere Application Server security for the Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service.

Once they are secured, to invoke any of the Web services provided by the Diameter Enabler, you need to include the WebSphere Application Server principal and credentials.

Diameter Enabler uses basic authentication to protect its Web services. It is recommended that you use Secure Sockets Layer (SSL)/Transport Layer Security (TLS) on your HTTP Transport chains if you want to have your Web service calls encrypted.

J2EE Access to internal Diameter Enabler infrastructure

Install the Diameter Web services (Rf, Ro, and Sh) on the same server as the Diameter Enabler base.

For a secure system, you should not install applications unless you are certain of their origin. Diameter Enabler extends the Channel Framework Architecture of WebSphere Application Server. Any application or plug-in to WebSphere Application Server has the ability to access a set of Diameter Enabler connections.

A rogue application could potentially bring up or down any of the Diameter Enabler connections, thus affecting the applications that use the connections. Therefore, an IMS Application Server application should not be installed on the same application server as the Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

Connections

You can use Transport Layer Security (TLS) or another security mechanism to create secure connections. TLS allows authentication for the client and server using certificates and encrypts the packets using the connection. If TLS is disabled, the connection must be secured with IPsec or by isolating your network so that it does not require transport or network layer security. Enabling TLS may affect performance.

Messages

A Diameter packet might pass through several hops to get to the final destination. After each hop in the path, the packet is decrypted and can be read by each device that receives it. The Base Diameter protocol (RFC 3588) does not provide end-to-end security. If your application requires end-to-end security, you can establish a direct connection between the client and server and use transport layer security or IPsec for this connection.

Configuration considerations for the WebSphere IMS Connector

The IBM WebSphere IP Multimedia Subsystem Connector provides interfaces and services used when developing applications that must interact with elements within the IMS core environment.

Overview of WebSphere IMS Connector

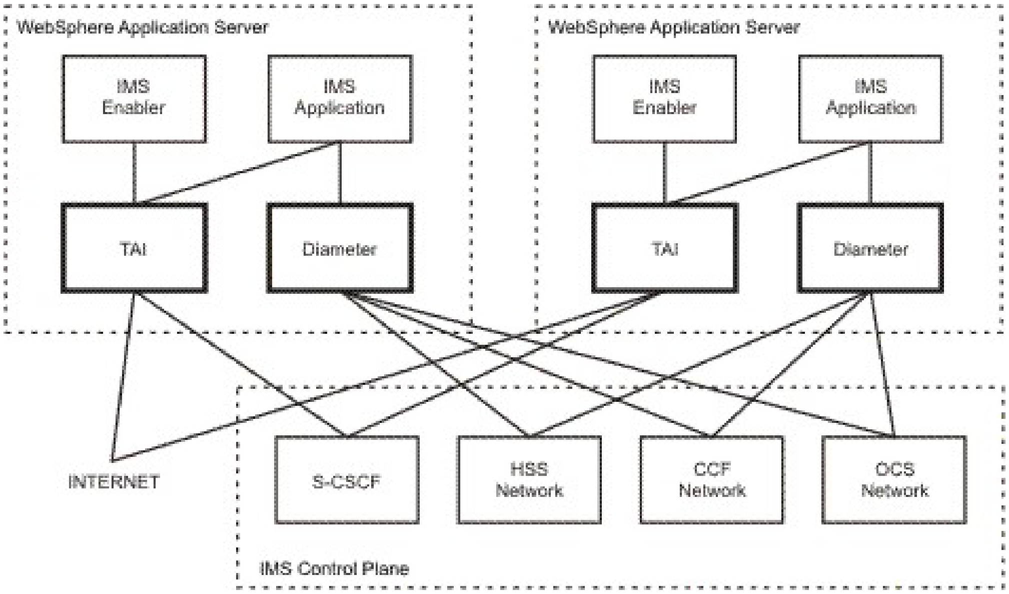
The IBM WebSphere IP Multimedia Subsystem Connector–consisting of the IBM WebSphere IMS Service Control Interfaces Component (ISC Interfaces), the Diameter Enabler, and the Trust Association Interceptor (TAI) security component–provides interfaces and services for use in developing applications that interact with elements in the IMS core environment.

The interfaces and services include the following:

- Serving-Call Session Control Function (S-CSCF)
- Home Subscriber Server (HSS)
- Charging functions: Online Charging System (OCS) and Charging Collection Function (CCF).

The WebSphere IMS Connector also provides authentication services and management services. The standard WebSphere IMS Connector configuration appears in the following diagram, where *Diameter* represents the Diameter Enabler, and *IMS Enabler* represents other IBM WebSphere software for Telecom products like the IBM WebSphere Presence Server and the IBM WebSphere XML Document Management Server Component.

Note: It is not necessary that all of these components reside on the same WebSphere Application Server instance.



ISC reference point support

Use of the WebSphere IMS Connector enables the application programmer to write applications that run in the WebSphere Application Server environment and interact with the IMS Control Plane through communication with an S-CSCF. This reference point between an application server and the S-CSCF is called *IMS Service Control* (ISC).

Because this reference point defines SIP as the protocol for communication, and because SIP is built into the base WebSphere Application Server product, there is no need to install additional code from the WebSphere IMS Connector. The Connector provides supporting services for such applications, and provides the license for future maintenance and support for such applications.

ISC Toolkit support

While it is not strictly a part of the WebSphere IMS Connector, the IBM WebSphere Telecom Toolkit provides sample applications. These applications demonstrate how to develop ISC applications that run in an application server with the WebSphere IMS Connector installed.

Diameter support

The IBM WebSphere Diameter Enabler Component, consisting of the Diameter Enabler base and Diameter Web services, is a major component of the WebSphere IMS Connector. The Diameter Enabler provides a Web service interface that allows Diameter applications to send and receive subscriber profile information, send accounting information, and manage online charging.

Security interface support

The WebSphere IMS Connector provides a TAI security component that provides integration for the IBM WebSphere software for Telecom products and other applications into the authentication protocols of the IMS core network. This component intercepts HTTP and SIP service requests to all IMS service plane components, verifies the sending element, consumes IMS-compliant identity information passed to it in HTTP/SIP headers, and propagates this information in a format compatible with WebSphere Application Server-based security.

Support for key performance indicators

The WebSphere IMS Connector includes the IBM Tivoli® Configuration Access Management (ITCAM) for J2EE Operations components for purposes of extracting and presenting key performance indicators (KPIs). This component is included as a convenience, and it provides the ability to integrate with a Tivoli or SNMP network manager. (The SNMP support is the Netcool®/System Service Monitors [SSM] 4.0 package.) Refer to the documentation provided as part of the product bundle.

Note: The following documentation is available for the installation and administration of the SNMP support:

- Netcool/SSM Release Notes
- Netcool/SSM Administration Guide
- Netcool/SSM Reference Guide
- Netcool/SSM PDF Documentation

For additional details about installing and removing Netcool fix packs, refer to the Netcool/SSM Patch Installation Guide.

For other updates and suggestions, search the IBM Technotes for Netcool/SSM 4.0 updates.

Chapter 5. Installing

IBM WebSphere Diameter Enabler Component has four subcomponents to install: Diameter Enabler base, Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service.

Diameter Enabler base must be manually installed into an existing WebSphere Application Server. All of the required files are packaged in the file `DHAImsConnectorInstallPackage_6.2.0.tar`. This package needs to be copied to *`was_root`* directory and unpacked on each node in Network Deployment environment. This places the required files in the appropriate directories.

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`

Install the IMS Connector product updates (fix packs and interim fixes) to the latest levels. Product fix packs are cumulative, so it is necessary to install only the latest available fix pack and any interim fixes that are available for that fix pack. Interim fix 1 will enable the WebSphere IMS Connector product to run on WebSphere Application Server version 7.0.0.1. Refer to the topic *Installing Updates* for details.

During the Diameter Enabler base installation, you will use the `DiameterChannelInstall.py` script to configure the TCP connections.

Note: The Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service are separate applications that can be installed and run independently or together. You need to install only the protocols that you plan to use.

The Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service are enterprise applications (EAR files) that are installed using the Integrated Solutions Console. You can find these installable EAR files in the following directory location: *`was_root`*`/installableApps/ImsConnector`.

If you are planning to use the subscribe and notify features of the Sh subscriber profile Web service or the Ro online charging Web service, you should install a database to store subscription information.

Enabling the application profiling service

If the underlying version of WebSphere Application Server is 7.x, you must enable the application profiling service for each application server that will run the WebSphere IMS Connector application.

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 **Servers** → **Server Types** → **WebSphere application servers**.
3. Click the name of the server where you plan to deploy the WebSphere IMS Connector application.
4. Expand Container Services and click **Application profiling service**.
5. Select the **Enable service at server startup** check box.
6. Click **OK** and to save your changes.
7. Repeat these steps for each server where you plan to deploy the WebSphere IMS Connector application.

What to do next

You are now ready to proceed with the installation of the full WebSphere IMS Connector product.

Standalone installation of WebSphere IMS Connector

Installing IBM WebSphere Diameter Enabler Component is a multiple step process. A standalone (non clustered) installation consists of preparing your environment, installing the subcomponents, and configuring Diameter Enabler base.

Preparing the environment

To install the Diameter Enabler successfully, you must first install and configure the prerequisite software. Also, you must first configure the application server platform and whichever of the following that you plan to use: Charging Collection Function (CCF), Online Charging System (OCS), and Home Subscriber Server (HSS).

Before you begin

Before beginning this task, you should review and verify that all hardware and software prerequisites have been met.

About this task

Complete the following steps to prepare the environment for the installation.

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`

1. Verify that your CCF, OCS, or HSS is configured properly to listen to the Diameter Enabler.
2. Open a command prompt.
3. Stop the application server.

4. 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.
5. Change (`cd`) to the *`was_root`* directory.
6. 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.

Preparing the database

Sh subscriber profile Web service and Ro online charging Web service require a database to store subscriber notification data for the subscribe and notify functions of the Web service. A database is required only if you plan to use the subscribe and notify capabilities of the Sh subscriber profile Web service or the subscribe/reauthorization notification capabilities of the Ro online charging Web service.

Preparing DB2

Create a DB2 database before installing Diameter Enabler.

Before you begin

Before you begin:

- DB2 should be installed and running

About this task

Some initial tuning values are provided; however, additional database tuning may be needed for optimal performance. Complete the following steps to create, configure, and connect to the database:

1. Log in to the DB2 server with a user ID that has database administrator authority, such as db2inst1.
2. Create a database named *diameter* by entering the following command from the command line:
   ```
   db2 "CREATE DATABASE diameter USING CODESET UTF-8 TERRITORY US
   COLLATE USING SYSTEM DFT_EXTENT_SZ 64"
   ```
3. To verify the database was created, type the following command:
   ```
   db2 "LIST DATABASE DIRECTORY"
   ```
   You should see a database entry for the `diameter` database.
4. Configure the database manager.
   ```
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MAXAGENTS 1000"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING SHEAPTHRES 20000"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MON_HEAP_SZ 512"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MAX_QUERYDEGREE 1"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_BUFPOOL off"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_LOCK off"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_SORT off"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_STMT off"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_TABLE off"
   ```

```
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_TIMESTAMP
off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_UOW off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING HEALTH_MON off"
```

5. Configure the database using the following series of commands:

```
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DBHEAP 8192"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING CATALOGCACHE_SZ
512"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGBUFSZ 2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING PCKCACHESZ
2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING APPLHEAPSZ
2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING APP_CTL_HEAP_SZ
4096"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING STAT_HEAP_SZ
8192"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING STMTHEAP 2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING SORTHEAP 512"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOCKLIST 10000"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING MAXLOCKS 45"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGFILSIZ 5000"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGPRIMARY 20"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGSECOND 20"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING MAXAPPLS 500"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AVG_APPLS 1"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING NUM_IOCLEANERS
3"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING NUM_IOSERVERS
3"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING SOFTMAX 100"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DFT_DEGREE 1"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DFT_PREFETCH_SZ
16"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DFT_EXTENT_SZ
64"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING CHNGPGS_THRESH
60"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AUTO_MAINT ON"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AUTO_TBL_MAINT
ON"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AUTO_RUNSTATS
ON"
```

6. Connect to the database and create all necessary tables using the following commands where *dbuser* represents the database user for the diameter database and *dbpassword* represents the password for the database user:

```
db2 "CONNECT TO diameter USER dbuser USING dbpassword"
db2 "create bufferpool BP_32K all nodes size 1000 pagesize 32K"
```

```
db2 "create regular tablespace SUBSCRIPTIONTS pagesize 32K managed by
system using ('SUBSCRIPTIONTS') extentsize 256 prefetchsize 64
bufferpool BP_32K"
```

```
db2 "create regular tablespace ROSUBSCRIPTIONTS pagesize 32K managed
by system using ('ROSUBSCRIPTIONTS') extentsize 256 prefetchsize 64
bufferpool BP_32K"
```

```
db2 "CREATE TABLE DIAMETERSHSUBSCRIPTION (SERVERNAME VARCHAR(250) NOT
NULL, PUBLICIDENTITY VARCHAR(250) NOT NULL, DATAREFERENCE INTEGER NOT
NULL, SERVICEINDICATION VARCHAR(250) NOT NULL, CALLBACKURL
VARCHAR(250) NOT NULL, DATESUBSCRIBED TIMESTAMP, USERID VARCHAR(250),
PASSWORD1 VARCHAR(250), SUBSCRIBEUSERDATA SMALLINT NOT NULL) IN
SUBSCRIPTIONTS"
```

```
db2 "ALTER TABLE DIAMETERSHSUBSCRIPTION ADD CONSTRAINT
PK_DIAMETERSHSUBS2 PRIMARY KEY (PUBLICIDENTITY, DATAREFERENCE,
CALLBACKURL, SUBSCRIBEUSERDATA, SERVICEINDICATION, SERVERNAME)"
```

```
db2 "CREATE TABLE DIAMETERROSUBSCRIPTION (SESSIONID VARCHAR(250) NOT
NULL,   DESTINATIONREALM VARCHAR(250) NOT NULL, CALLBACKURI
VARCHAR(250), DATESUBSCRIBED TIMESTAMP, DATEEXPIRESSECONDS BIGINT NOT
NULL, USERID VARCHAR(250), PASSWORD1 VARCHAR(250)) IN
ROSUBSCRIPTIONTS"
```

```
db2 "ALTER TABLE DIAMETERROSUBSCRIPTION ADD CONSTRAINT
PK_DIAMETERROSUBS2 PRIMARY KEY (SESSIONID, DESTINATIONREALM)"
```

7. Restart DB2 using the following commands:

   ```
   db2stop force
   ```

   ```
   db2start
   ```

Preparing Oracle

Create an Oracle database user and tables before installing Diameter Enabler.

Before you begin

Before you begin:

- Oracle should be installed and running.
- Oracle database should be created, and is referred to in the following procedure as *diameter*.

About this task

Some initial tuning values are provided; however, additional database tuning may be needed for optimal performance. Complete the following steps to create the appropriate user ID and database table:

1. Log in to the Oracle server with a user ID that has database administrator authority, such as oracle.
2. Start SQLPlus and connect to the *diameter* database with SYSTEM authority.
3. Create an ID for the Diameter Enabler to access the database. The following commands create the user and gives the user the appropriate privileges:

   ```
   create user diameter_user identified by password;
   ```

   ```
   grant create session, create table, create sequence,
   exp_full_database, imp_full_database, unlimited tablespace to
   diameter_user;
   ```

   Where:

`diameter_user` is the new Oracle user ID Diameter Enabler will use to access the database.

`password` is the password for the *diameter_user*.

4. Verify the correct users are created and have the correct privileges by typing the following command in SQLPlus:

   ```
   select username from DBA_USERS;
   ```

   This command will list the existing users that have database privileges.
5. Connect to the database with *diameter_user* using the following command:

   ```
   connect diameter_user/password@diameter
   ```

   Where:

   `diameter_user` is the new Oracle user ID Diameter Enabler will use to access the database.

   `password` is the password for the *diameter_user*.

   `diameter` is the database name
6. Create the `DIAMETERSHSUBSCRIPTION` database table and the primary key using the following commands:

   ```
   create table DIAMETERSHSUBSCRIPTION ( SERVERNAME VARCHAR2(250) NOT
   NULL, PUBLICIDENTITY VARCHAR2(250) NOT NULL, DATAREFERENCE INT NOT
   NULL, SERVICEINDICATION VARCHAR2(250) NOT NULL, CALLBACKURL
   VARCHAR2(250) NOT NULL, DATESUBSCRIBED DATE, USERID VARCHAR2(250),
   PASSWORD VARCHAR2(250), SUBSCRIBEUSERDATA SMALLINT NOT NULL ) PCTFREE
   0 PCTUSED 0 LOGGING;
   ```

   ```
   ALTER TABLE DIAMETERSHSUBSCRIPTION ADD CONSTRAINT
   PK_DIAMETERSHSUBSCRIPTION PRIMARY KEY (PUBLICIDENTITY, DATAREFERENCE,
   CALLBACKURL, SUBSCRIBEUSERDATA, SERVICEINDICATION, SERVERNAME);
   ```
7. Create the `DIAMETERROSUBSCRIPTION` database table and the primary key using the following commands:

   ```
   create table DIAMETERROSUBSCRIPTION ( SESSIONID VARCHAR2(250) NOT
   NULL, DESTINATIONREALM VARCHAR2(250) NOT NULL,  CALLBACKURI
   VARCHAR2(250), DATESUBSCRIBED DATE,  DATEEXPIRESSECONDS NUMBER(19 ,
   0) NOT NULL, USERID VARCHAR2(250), PASSWORD VARCHAR2(250) ) LOGGING;
   ```

   ```
   ALTER TABLE DIAMETERROSUBSCRIPTION ADD CONSTRAINT
   PK_DIAMETERROSUBSCRIPTION PRIMARY KEY (SESSIONID, DESTINATIONREALM);
   ```

What to do next

To improve performance, you may need to create a separate table space for the DIAMETERSHSUBSCRIPTION and DIAMETERROSUBSCRIPTION tables and complete additional database tuning.

Installing Diameter Enabler base (Standalone)

Diameter Enabler base is installed into the `plugins` directory in your WebSphere Application Server Network Deployment path. The installation process includes minor configuration steps necessary to run the Diameter Enabler base.

Before you begin

Before you begin, the following software should be installed:

- 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.

- If you require subscribe and notify functions for either Ro or Sh, one of the following databases for storing subscriber notification data:
  - DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2
  - Oracle Oracle Database, version 10.2.0.2

In addition, you should have:

- Completed the environment preparation steps, which includes unpacking the `DHAImsConnectorInstallPackage_6.2.0.tar` file into the *`was_root`* 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`
- Prepared the database

You will need the following files for this installation task:

- `com.ibm.ws.diameter_6.2.0.jar` (in *`was_root`*`/plugins`)
- `DiameterChannelInstall.py` (in *`was_root`*`/installableApps/ImsConnector/install`)

You will need the following information to complete this task:

- WebSphere Application Server user name and password. You only need this information if security is enabled.
- Cell name and node name where the WebSphere Application Server server is installed; the default server is server1.
- Fully qualified host name where the node is installed.
- Port number for the Diameter inbound channel. Port 3868 is the preferred port defined in RFC 3588. You should use this port unless it is already in use.

Tip:

- To locate your host name and node name, refer to the `AboutThisProfile.txt` in the following node profile directory:

  AIX *`was_profile_root`*`/logs/`

  Linux *`was_profile_root`*`/logs/`

  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`*

  For example, AppSrv01 in a standalone environment.
- To locate your cell name, refer to the directory name found in the following location:

  AIX *`was_profile_root`*`/config/cells/`*`cell_name`*

  Linux *`was_profile_root`*`/config/cells/`*`cell_name`*

  Where:

  cell_name is your actual cell name

About this task

Complete the following steps to install the Diameter Enabler base:

1. Log in to the server where WebSphere Application Server is installed.
2. If you have not already done so, unpack the `DHAImsConnectorInstallPackage_6.2.0.tar` file as described in preparing the environment. Otherwise, open a command prompt and verify that the file `com.ibm.ws.diameter_6.2.0.jar` was correctly copied to the `plugins` directory:

   AIX Linux *`was_root`*`/plugins`
3. Start the application server.
4. Configure the Diameter channels. Run the following command:

   Important: Enter the following parameters on a single line.

   AIX Linux
   ```
   was_profile_root/bin/wsadmin.sh -username user_name -password
   password -f script_path/DiameterChannelInstall.py cell_name node_name
   server_name host_name port_number standalone [debug]
   ```
   Where:

   `user_name` represents your WebSphere Application Server user ID. This parameter is required if security is enabled.

   `password` represents the password associated with your *`user_name`*. This parameter is required if security is enabled.

   `script_path` represents the path to `DiameterChannelInstall.py`. After the unpack operation, *script_path* is *`was_root`*`/installableApps/ImsConnector/install`.

   `cell_name` represents the name of cell where the server is installed

   `node_name` represents the name of node where the server is installed

   `server_name` represents the name of the application server where you are installing Diameter Enabler base

   `host_name` represents the fully qualified host name the Diameter TCP channel will bind to. To specify any host name, type `"*"` (including the quotes) for the value.

   `port_name` represents the port number for the Diameter inbound TCP channel, 3868 preferred

   `standalone` indicates that the script is running in a standalone environment

   `debug` enables debugging for the configuration script (optional)
5. Stop the server where you have installed Diameter Enabler base.
6. Start the server.

Verifying the Diameter Enabler base installation (Standalone)

Once the Diameter Enabler base is installed into the `plugins` directory, you should verify the installation.

Before you begin

Before you can verify the installation, you must have completed the installation process.

About this task

Complete the following steps to verify the installation of the Diameter Enabler base:

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 **Servers** → **Application servers**.
3. Click ***server_name***.
4. Under Communications, click **Ports**.
5. Verify the end point is properly configured:
   a. Verify the **DiameterNamedEndPoint** appears in the list and has the following values:

      Host: * or *host_name*

      Port: *3868* (This is the port number you defined for the Diameter inbound channel.)
6. Verify the DiameterChain is properly configured:
   a. Click **View associated transports** on the **DiameterNamedEndPoint** row.
   b. Verify **DiameterChain** appears in the list and has the following values:

      Enabled: Enabled

      Host: * or *host_name*

      Port: *3868* (This is the port number you defined for the Diameter inbound channel.)

      SSL Enabled: Disabled
7. Verify the **SecureDiameterChain** appears in the list and is properly configured with the following values:

   Enabled: Enabled

   Host: * or *host_name*

   Port: *3868* (This is the port number you defined for the Diameter inbound channel.)

   SSL Enabled: Enabled
8. Verify the DiameterChain channel is properly configured:
   a. Click **DiameterChain**.
   b. Verify the following information is correct in the **Transport Channels** section:
      - TCP inbound channel (DiameterTCPInboundChannel)
        - Host: * or *server_name*
        - Port: *3868* (This is the port number you defined for the Diameter inbound channel.)
        - Thread pool: DiameterThreadPool
      - Generic inbound channel (DiameterGenericInboundChannel)

c. Click **Generic inbound channel (DiameterGenericInboundChannel)**.

d. Verify the following values are correct:

   Transport Channel Name: DiameterGenericInboundChannel

   Discrimination weight: 10

   JAR file: com.ibm.ws.diameter_6.2.0.jar

   Channel type identifier: DiameterInboundChannel

   Configuration URI should be empty.

e. Click **Cancel** on the DiameterChain panel, and click **Cancel** to return to the Transport chain panel.

9. Verify the SecureDiameterChain channel is properly configured:
   a. Click **SecureDiameterChain**.
   b. Verify the following information is correct in the **Transport Channels** section:
      - TCP inbound channel (DiameterTCPInboundChannel)
        - Host: * or *server_name*
        - Port: *3868* (This is the port number you defined for the Diameter inbound channel.)
        - Thread pool: DiameterThreadPool
      - SSL inbound channel (DiameterSSLInboundChannel)
        - SSL Configuration: Diameter
      - Generic inbound channel (SecureDiameterGenericInboundChannel)
   c. Click **Generic inbound channel (SecureDiameterGenericInboundChannel)**.
   d. Verify the following values are correct:

      Transport Channel Name: SecureDiameterGenericInboundChannel

      Discrimination weight: 1

      JAR file: com.ibm.ws.diameter_6.2.0.jar

      Channel type identifier: DiameterInboundChannel

      Configuration URI should be empty.
   e. Click **Cancel** on the SecureDiameterChain panel, and click **Cancel** to return to the Transport chain panel.
10. Verify the SSL configuration object has been created and configured properly:
    a. In the navigation pane, click **Security** → **SSL certificate and key management** → **SSL configurations**.
    b. Click **Diameter** in the list of SSL configurations.
    c. Under Additional Properties, click **Quality of protection (QoP) settings**.
    d. Verify the following values are correct:

       Client authentication: Required

       Protocol: SSL_TLS
    e. Click **Cancel**.

Connecting to the database

After the database and WebSphere Application Server Network Deployment are installed, you must create the data source in the application server for the Sh subscriber profile Web service and Ro online charging Web service to be able to access the database for the subscribe and notification functions. To connect to the database, you must create a JAAS authentication alias for the database, create the JDBC provider, and define the data source using the Integrated Solutions Console.

The database client allows the database server and the application server to communicate. When the database is not on the same server as WebSphere Application Server Network Deployment, the JDBC JAR file (or the full database client) must be installed on each WebSphere Application Server Network Deployment server. This applies to all nodes.

Creating an authentication alias

Use an authentication alias to define authentication data used to access the database. The authentication alias is created using the Integrated Solutions Console.

Before you begin

Before you begin, the following software should be installed:

- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Started the application server
- Created the database

About this task

Complete the following steps to create an authentication alias for the Diameter database:

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.

   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. Expand **Java Authentication and Authorization Service**, and click **J2C authentication data**.
4. Click **New**.
5. In the **Alias** field, type `diameter_alias`. Note of the name of the alias; you will need it later.
6. In the **User ID** field, type the database administrator user ID that can be used to access the Diameter database. This is the same user ID used in the *Preparing the database* topics.
7. In the **Password** field, type the password that corresponds to the user ID.
8. Optional: In the **Description** field, type a description for the alias.
9. Click **OK**.
10. Click **Save** to save changes to the master configuration.

Creating a JDBC provider (Standalone)

To access a database, you must first create a JDBC provider in the Integrated Solutions Console.

Before you begin

Before you begin, the following software should be installed:

- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Started the application server
- Created the database
- Created an authentication alias for the database

About this task

Complete the following steps to create a JDBC provider:

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 **Resources** → **JDBC** → **JDBC Providers**.

3. Select *cell_name* from the Scope drop-down list.
4. Click **New**.
5. Create the JDBC provider:
   a. Select your database type from the drop-down list.
      - DB2 DB2
      - Oracle Oracle
   b. Select the **Provider type** for your database:
      - DB2 DB2 Universal JDBC Driver Provider
      - Oracle Oracle JDBC Driver
   c. Select **Connection pool data source** for the **Implementation type**.
   d. Click **Next**.
6. Enter the value for the Directory location for the database client JAR files, which is saved as a WebSphere variable. For the path, use the following format where `db_client_root` is the path where you installed the database client:
   - DB2 `/db_client_root/java`
   - Oracle `/db_client_root/jdbc/lib`

   This path is where you would find the following JAR files:
   - DB2 db2jcc.jar and db2jcc_license_*.jar
   - Oracle classes*.jar and ojdbc14*.jar

   Note: If the location ever requires updating, make sure that the DB2UNIVERSAL_JDBC_DRIVER_PATH or ORACLE_JDBC_DRIVER_PATH variable is updated at both the cell and node scope from **Environment** → **WebSphere Variables** in the Integrated Solutions Console.

   Click **Next**.
7. Verify that all values are correct, and click **Finish**.
8. Click **Save** to save changes to the master configuration.
9. Click **OK** when node synchronization has completed.

Defining the data source

Data sources are the resources that provide connections to your relational database. Use the Integrated Solutions Console to define data sources.

Before you begin

Before you begin, the following software should be installed:
- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, the following steps should be completed:
- Started the application server

- Created the database
- Created WebSphere variables
- Created an authentication alias for the database
- Created the JDBC provider

About this task

Complete the following steps to define the data source to map the connection to the Diameter database:

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 **Resources** → **JDBC** → **JDBC Providers**.
3. Click the name of the JDBC provider for which you are configuring the data source, to display its properties. The JDBC provider was created in the previous task.
4. Under Additional Properties, click **Data sources**.
5. Click **New**.
6. Type `Diameter` in the **Data source name** field.
7. Type `jdbc/diameter` in the **JNDI name** field.
8. Select ***diameter_alias*** from the **Component-managed authentication alias** drop-down list, and click **Next**.
9. Oracle Configure the database properties:
   a. Type `jdbc:oracle:thin:@`*`host_name`*`:`*`port_number`*`:`*`database_sid`* in the **URL** field. Where:

      host_name represents the host name of the server where Oracle Database is installed

      port_number represents the port number used to access the database server (1521 is the default port)

      database_sid represents the system identifier for the database you created
   b. Select **Oracle10g data store helper** from the **Data store helper class name** drop-down list.
   c. Click **Next**.
10. DB2 Configure the database properties:
    a. Type *`database_name`* in the **Database name** field. This is the name of the database you created.

   b. Click **4** in the **Driver type** drop-down list to specify the connectivity type of the data source. This value corresponds with the driver type property in the data source class.
   c. Type `server_name` in the **Server name** field. This is the fully qualified host name for the DB2 server.
   d. Type `port_number` in the **Port number** field. This is the port the DB2 server is listening on. (Port 50000 is the default port.)
   e. Click **Next**.

11. Select the **Use this data source in container managed persistence (CMP)** check box. Click **Next**.
12. Verify that the values are correct, and click **Finish**.
13. Click **Save** to save changes to the master configuration.
14. Click **OK** when node synchronization has completed.
15. Restart the application server.
   a. Stop the application server. 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
      ```

      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. Start the application server. 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
      ```

      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*.

Testing the database connection

After creating the data source, use the Integrated Solutions Console to test the connection to the database.

Before you begin

Note: This procedure should only be used if the database client or JDBC JAR files are present, which may not be the case on the deployment manager.

Before you begin, the following software should be installed:

- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, you should have already completed the following steps:

- Started the application server
- Created the database
- Created WebSphere variables
- Created an authentication alias for the database
- Created the JDBC provider
- Defined the data source

About this task

Complete the following steps to test the mapped database connection:

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 **Resources** → **JDBC** → **JDBC Providers**.
3. Click the name of the JDBC provider for the component you are installing.
4. Under Additional Properties, click **Data sources**.
5. Select the associated check box for the data source to test the connection.
6. Click **Test connection**. A message similar to the following indicates a successful connection:

   ```
   Test connection for data source datasource_name on server server_name at node node_name was succe
   ```

   Where:

   datasource_name represents the name of the data source you created

   server_name represents the name of the server where you created the data source *node_name* represents the node that contains *server_name*

Installing the services

Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service are deployed as enterprise applications on your WebSphere Application Server Network Deployment node.

Installing Rf accounting Web service (Standalone)

Rf accounting Web service is a messaging interface to enable an application to send accounting messages to a Charging Collection Function (CCF).

Before you begin

Before you begin, the following software should be installed:

- 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.
- Diameter Enabler base

Verify that you have the following files needed for the installation:

- DHADiameterRfWebServiceEAR (in `*was_root*/installableApps/ImsConnector`)

  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`
- `Diameter_Rf.properties` (in *`was_root`*`/installableApps/ImsConnector/properties`)

1. Log in to the server where WebSphere Application Server is installed.
2. Copy `Diameter_Rf.properties` to the following directory:

   AIX Linux *`was_profile_root`*`/properties`

   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`*

   For example, AppSrv01.
3. Open `Diameter_Rf.properties` in a text editor.
4. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
5. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
6. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a

multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.

7. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

   The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.
8. 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**.
9. Install DHADiameterRfWebServiceEAR by completing the following steps:
   a. In the navigation panel, click **Applications** → **Install new application**.
   b. Click **Browse** to locate DHADiameterRfWebServiceEAR.
   c. Select **Show me all installation options and parameters**.
   d. Click **Next** → **Next** → **Continue**.
   e. If security is enabled, click **Step 7: Map security roles to users or groups**, and select the check box corresponding to **Diameter_Role** to map the user and groups that you wish to grant access to the Rf accounting Web service.
   f. Click **Step 8: Summary**, and verify all options have the correct values.
   g. Click **Finish**.
   h. Click **Save** to save changes to the master configuration.
10. In the navigation panel, click **Applications** → **Enterprise applications**.
11. Select the check box corresponding to **DHADiameterRfWebServiceEAR**.
12. Click **Start**. You should receive the following message: Application DHADiameterRfWebServiceEAR on server *server_name* and node *node_name* started successfully.
13. Verify **Application status** for DHADiameterRfWebServiceEAR is started. After the application is started, Diameter Enabler base will start to establish connections.

Installing Ro online charging Web service (Standalone)

Ro online charging Web service is a messaging interface to enable an application to send credit control messages to an Online Charging System (OCS).

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following supported databases:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Completed the environment preparation steps.
- Started the application server.
- Connected to the database.

Verify that you have the following files needed for the installation:

- DHADiameterRoWebServiceEAR (in *was_root*`/installableApps/ImsConnector`)

  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`
- `Diameter_Ro.properties` (in *was_root*`/installableApps/ImsConnector/properties`)

1. Log in to the server where WebSphere Application Server is installed.
2. Copy `Diameter_Ro.properties` to the following directory:

   AIX Linux *was_profile_root*`/properties`

   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*

   For example, AppSrv01.
3. Open `Diameter_Ro.properties` in a text editor.
4. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
5. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
6. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a

multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.

7. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

   The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.
8. 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**.
9. Install DHADiameterRoWebServiceEAR by completing the following steps:
   a. In the navigation panel, click **Applications** → **Install new application**.
   b. Click **Browse** to locate DHADiameterRoWebServiceEAR.
   c. Select **Show me all installation options and parameters**.
   d. Click **Next** → **Next** → **Continue**.
   e. Optional: If you are using the subscribe and notification functions, click **Step 3: Provide options to perform the EJB Deploy**.
      - **Deploy EJB option - Database type**: Choose your database:

        DB2 DB2UDB_V91

        Oracle ORACLE_V10G
      - **Database schema**: Type your database user name. This is the same user name used in *Preparing the database* section; for example, if you are using the default DB2 user name, type `db2inst1`.
   f. Optional: If you are using the subscribe and notification functions, click **Step 9: Map data sources for all 2.x CMP beans**, and click **Browse** to select the **Target Resource JNDI Name** for the module. The JNDI name was created in the *Defining the data source* section.
   g. If security is enabled, click **Step 13: Map security roles to users or groups**, and select the check box corresponding to **Diameter_Role** to map the user and groups that you wish to grant access to the Ro online charging Web service.
   h. If security is enabled, click **Step 14: Map RunAs roles to users**, and map a user that is assigned the Diameter_Role to the RunAs role. The RoNotification EJB will run as this user when a notification is received.

Note: Any future password changes need to be updated in this enterprise application's settings as well.

   i. Click **Step 15: Summary**, and verify all options have the correct values.
   j. Click **Finish**.
   k. Click **Save** to save changes to the master configuration.

10. In the navigation panel, click **Applications** → **Enterprise applications**.
11. Select the check box corresponding to **DHADiameterRoWebServiceEAR**.
12. Click **Start**. You should receive the following message: Application DHADiameterRoWebServiceEAR on server *server_name* and node *node_name* started successfully.
13. Verify **Application status** for DHADiameterRoWebServiceEAR is started. After the application is started, Diameter Enabler base will start to establish connections.

Installing Sh subscriber profile Web service (Standalone)

Sh subscriber profile Web service is used by an application server to download user profile data from the Home Subscriber Server.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following supported databases:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Completed the environment preparation steps.
- Started the application server.
- Connected to the database.

Verify that you have the following files needed for the installation:

- DHADiameterShWebServiceEAR (in *was_root*`/installableApps/ImsConnector`)

  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`
- `Diameter_Sh.properties` (in *was_root*`/installableApps/ImsConnector/properties`)
- `ShDataType.xsd` (in *was_root*`/installableApps/ImsConnector/xsd`)

1. Log in to the server where WebSphere Application Server is installed.
2. Copy `ShDataType.xsd` to the following directory:

   AIX Linux *was_profile_root*

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`*

For example, AppSrv01.

3. Copy `Diameter_Sh.properties` to the following directory:

   AIX Linux *`was_profile_root`*`/properties`
4. Open `Diameter_Sh.properties` in a text editor.
5. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
6. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
7. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.
8. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

   The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.
9. 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**.
10. Install DHADiameterShWebServiceEAR by completing the following steps:
    1. In the navigation panel, click **Applications** → **Install new application**.
    2. Click **Browse** to locate DHADiameterShWebServiceEAR.
    3. Select **Show me all installation options and parameters**.
    4. Click **Next** → **Next** → **Continue**.

e. Optional: If you are using the subscribe and notification functions, click **Step 3: Provide options to perform the EJB Deploy**.
   - **Deploy EJB option - Database type**: Choose your database:

     DB2 DB2UDB_V91

     Oracle ORACLE_V10G
   - **Database schema**: Type your database user name. This is the same user name used in *Preparing the database* section; for example, if you are using the default DB2 user name, type `db2inst1`.

f. Optional: If you are using the subscribe and notification functions, click **Step 9: Map data sources for all 2.x CMP beans**, and click **Browse** to select the **Target Resource JNDI Name** for the module. The JNDI name was created in the *Defining the data source* section.

g. If security is enabled, click **Step 13: Map security roles to users or groups**, and select the check box corresponding to **Diameter_Role** to map the user and groups that you wish to grant access to the Sh subscriber profile Web service.

h. If security is enabled, click **Step 14: Map RunAs roles to users**, and map a user that is assigned the Diameter_Role to the RunAs role. The ShNotification EJB will run as this user when a notification is received.

   Note: Any future password changes need to be updated in this enterprise application's settings as well.

i. Click **Step 15: Summary**, and verify all options have the correct values.

j. Click **Finish**.

k. Click **Save** to save changes to the master configuration.

11. In the navigation panel, click **Applications** → **Enterprise applications**.
12. Select the check box corresponding to **DHADiameterShWebServiceEAR**.
13. Click **Start**. You should receive the following message: Application DHADiameterShWebServiceEAR on server *server_name* and node *node_name* started successfully.
14. Verify **Application status** for DHADiameterShWebServiceEAR is started. After the application is started, Diameter Enabler base will start to establish connections.

Verifying the connection

After you complete the installation, you should verify that the Diameter Enabler is able to connect with other Diameter peers. Diameter peers include the Charging Collection Function (CCF), Online Charging System (OCS), Home Subscriber Server (HSS), and agents.

1. Verify the TCP listener port is active by typing the following command:

   ```
   netstat -na
   ```

   You should see a list of active listening ports, including 3868, the default listening port for Diameter Enabler base.

   ```
   tcp      0      0 0.0.0.0:3868     0.0.0.0:*     LISTEN
   ```
2. Verify the connection is established between the Diameter peer and the Diameter Enabler base by reissuing the netstat command.

   When the Diameter Enabler base initiates a connection with a Diameter peer, you will see the following result:

   ```
   tcp      1     0 9.42.125.21:32861     209.132.177.100:3868     ESTABLISHED
   ```

Where:

`9.42.125.21` represents the IP address of the Diameter Enabler base

`32861` represents the ephemeral source port from theDiameter Enabler base, which will remain the same as long as the connection is active. If configurable source port is enabled, then it will show that port number instead.

`209.132.177.100` represents the IP address of the Diameter peer

`3868` represents the listener port of the Diameter peer

When the Diameter peer initiates a connection with Diameter Enabler base, you will see the following result:

```
tcp      1     0 9.42.125.21:3868     209.132.177.100:43125     ESTABLISHED
```

Where:

`9.42.125.21` represents the IP address of the Diameter Enabler base

`3868` represents the listener port of the Diameter Enabler base

`209.132.177.100` represents the IP address of the Diameter peer

`43125` represents the ephemeral source port from the Diameter peer, which will remain the same as long as the connection is active

Clustered installation of IBM WebSphere Diameter Enabler Component

IBM WebSphere Diameter Enabler Component utilizes horizontal and vertical clustering capabilities provided with WebSphere Application Server Network Deployment. Using clustering techniques, you can achieve a high-availability system environment.

Scaling may be vertical only, horizontal only, or a combination. For both horizontal and vertical clusters, each server must have a unique OriginHostName and a unique HostIpAddress and port number pair.

Horizontal cluster

Horizontal clusters are servers on separate physical servers. Each server must have a unique IP address. Each server may be configured to listen on the same port number. 3868 is the recommended port number because the RFC references that port for Diameter connections.

Vertical cluster

Vertical clusters are application servers on the same physical server. Each application server may have the same IP address. The IP address and TCP port combination must be unique for every cluster member. If the IP address is the same for two or more vertical cluster members, the port must be different.

Before beginning a clustered installation, the following tasks must be complete:

- Installed and configured all prerequisite software for the environment
- Copied the required installation files onto the appropriate application server
- Created a deployment manager profile
- Added (federated) all nodes into the deployment manager cell

For the Diameter Enabler clustered installation procedure, the documentation assumes you have already created a deployment manager profile and federated all nodes into the deployment manager cell. For this installation path, you install and configure Diameter Enabler base. Then, you create the cluster and install the Web services applications into the cluster.

Additionally, you must have a proxy server for Diameter Enabler. Additional information for the proxy server is referenced when you create the cluster in the procedure.

The high-level steps for the clustered installation procedure are as follows:

Preparing the environment (Cluster)

To install the Diameter Enabler successfully, you must first install and configure the prerequisite software. You must also configure the application server platform, along with whichever of the following that you plan to use, for a clustered installation: Charging Collection Function (CCF), Online Charging System (OCS), and Home Subscriber Server (HSS).

Before you begin

Before beginning the installation, you must have completed the following tasks on WebSphere Application Server Network Deployment:

- Created a deployment manager profile
- Federated all nodes into the deployment manager cell

About this task

Complete the following steps to prepare the environment for the installation.

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`

1. Verify that your CCF, OCS, or HSS is configured properly to listen to the Diameter Enabler.
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. Verify that the console preferences are set to synchronize changes across the nodes:
    a. In the navigation panel, click **System administration** → **Console Preferences**.
    b. Select **Synchronize changes with Nodes**.

   c. Click **Apply**.
4. Stop the deployment manager.
5. Stop the node agent for each federated node.
6. Unpack the installation tar file to the *`was_root`* directory on the deployment manager and on each federated node. Perform the following steps:

   Note: If these are on separate machines, repeat these steps for each machine.

   a. Log in to the server where the deployment manager or federated node is installed.
   b. Copy the installation file from the CD (`IBM_WebSphere_IMS_Connector/DHAImsConnectorInstallPackage_6.2.0.tar`) to the *`was_root`* directory.
   c. Change (`cd`) to the *`was_root`* directory.
   d. Unpack the file by typing the following command: `tar -pxvf DHAImsConnectorInstallPackage_6.2.0.tar`

Preparing the database

Sh subscriber profile Web service and Ro online charging Web service require a database to store subscriber notification data for the subscribe and notify functions of the Web service. A database is required only if you plan to use the subscribe and notify capabilities of the Sh subscriber profile Web service or the subscribe/reauthorization notification capabilities of the Ro online charging Web service.

Preparing DB2

Create a DB2 database before installing Diameter Enabler.

Before you begin

Before you begin:
- DB2 should be installed and running

About this task

Some initial tuning values are provided; however, additional database tuning may be needed for optimal performance. Complete the following steps to create, configure, and connect to the database:

1. Log in to the DB2 server with a user ID that has database administrator authority, such as db2inst1.
2. Create a database named *diameter* by entering the following command from the command line:
   ```
   db2 "CREATE DATABASE diameter USING CODESET UTF-8 TERRITORY US
   COLLATE USING SYSTEM DFT_EXTENT_SZ 64"
   ```
3. To verify the database was created, type the following command:
   ```
   db2 "LIST DATABASE DIRECTORY"
   ```
   You should see a database entry for the `diameter` database.
4. Configure the database manager.
   ```
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MAXAGENTS 1000"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING SHEAPTHRES 20000"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MON_HEAP_SZ 512"
   db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MAX_QUERYDEGREE 1"
   ```

```
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_BUFPOOL off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_LOCK off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_SORT off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_STMT off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_TABLE off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_TIMESTAMP
off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING DFT_MON_UOW off"
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING HEALTH_MON off"
```

5. Configure the database using the following series of commands:

```
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DBHEAP 8192"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING CATALOGCACHE_SZ
512"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGBUFSZ 2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING PCKCACHESZ
2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING APPLHEAPSZ
2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING APP_CTL_HEAP_SZ
4096"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING STAT_HEAP_SZ
8192"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING STMTHEAP 2048"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING SORTHEAP 512"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOCKLIST 10000"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING MAXLOCKS 45"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGFILSIZ 5000"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGPRIMARY 20"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING LOGSECOND 20"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING MAXAPPLS 500"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AVG_APPLS 1"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING NUM_IOCLEANERS
3"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING NUM_IOSERVERS
3"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING SOFTMAX 100"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DFT_DEGREE 1"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DFT_PREFETCH_SZ
16"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING DFT_EXTENT_SZ
64"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING CHNGPGS_THRESH
60"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AUTO_MAINT ON"
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AUTO_TBL_MAINT
ON"
```

```
db2 "UPDATE DATABASE CONFIGURATION FOR diameter USING AUTO_RUNSTATS
ON"
```

6. Connect to the database and create all necessary tables using the following commands where *dbuser* represents the database user for the diameter database and *dbpassword* represents the password for the database user:

   ```
   db2 "CONNECT TO diameter USER dbuser USING dbpassword"
   ```

   ```
   db2 "create bufferpool BP_32K all nodes size 1000 pagesize 32K"
   ```

   ```
   db2 "create regular tablespace SUBSCRIPTIONTS pagesize 32K managed by
   system using ('SUBSCRIPTIONTS') extentsize 256 prefetchsize 64
   bufferpool BP_32K"
   ```

   ```
   db2 "create regular tablespace ROSUBSCRIPTIONTS pagesize 32K managed
   by system using ('ROSUBSCRIPTIONTS') extentsize 256 prefetchsize 64
   bufferpool BP_32K"
   ```

   ```
   db2 "CREATE TABLE DIAMETERSHSUBSCRIPTION (SERVERNAME VARCHAR(250) NOT
   NULL, PUBLICIDENTITY VARCHAR(250) NOT NULL, DATAREFERENCE INTEGER NOT
   NULL, SERVICEINDICATION VARCHAR(250) NOT NULL, CALLBACKURL
   VARCHAR(250) NOT NULL, DATESUBSCRIBED TIMESTAMP, USERID VARCHAR(250),
   PASSWORD1 VARCHAR(250), SUBSCRIBEUSERDATA SMALLINT NOT NULL) IN
   SUBSCRIPTIONTS"
   ```

   ```
   db2 "ALTER TABLE DIAMETERSHSUBSCRIPTION ADD CONSTRAINT
   PK_DIAMETERSHSUBS2 PRIMARY KEY (PUBLICIDENTITY, DATAREFERENCE,
   CALLBACKURL, SUBSCRIBEUSERDATA, SERVICEINDICATION, SERVERNAME)"
   ```

   ```
   db2 "CREATE TABLE DIAMETERROSUBSCRIPTION (SESSIONID VARCHAR(250) NOT
   NULL,   DESTINATIONREALM VARCHAR(250) NOT NULL, CALLBACKURI
   VARCHAR(250), DATESUBSCRIBED TIMESTAMP, DATEEXPIRESSECONDS BIGINT NOT
   NULL, USERID VARCHAR(250), PASSWORD1 VARCHAR(250)) IN
   ROSUBSCRIPTIONTS"
   ```

   ```
   db2 "ALTER TABLE DIAMETERROSUBSCRIPTION ADD CONSTRAINT
   PK_DIAMETERROSUBS2 PRIMARY KEY (SESSIONID, DESTINATIONREALM)"
   ```

7. Restart DB2 using the following commands:

   ```
   db2stop force
   ```

   ```
   db2start
   ```

Preparing Oracle

Create an Oracle database user and tables before installing Diameter Enabler.

Before you begin

Before you begin:

- Oracle should be installed and running.
- Oracle database should be created, and is referred to in the following procedure as *diameter*.

About this task

Some initial tuning values are provided; however, additional database tuning may be needed for optimal performance. Complete the following steps to create the appropriate user ID and database table:

1. Log in to the Oracle server with a user ID that has database administrator authority, such as oracle.
2. Start SQLPlus and connect to the *diameter* database with SYSTEM authority.

3. Create an ID for the Diameter Enabler to access the database. The following commands create the user and gives the user the appropriate privileges:

   ```
   create user diameter_user identified by password;
   ```

   ```
   grant create session, create table, create sequence,
   exp_full_database, imp_full_database, unlimited tablespace to
   diameter_user;
   ```

   Where:

   `diameter_user` is the new Oracle user ID Diameter Enabler will use to access the database.

   `password` is the password for the *diameter_user*.
4. Verify the correct users are created and have the correct privileges by typing the following command in SQLPlus:

   ```
   select username from DBA_USERS;
   ```

   This command will list the existing users that have database privileges.
5. Connect to the database with *diameter_user* using the following command:

   ```
   connect diameter_user/password@diameter
   ```

   Where:

   `diameter_user` is the new Oracle user ID Diameter Enabler will use to access the database.

   `password` is the password for the *diameter_user*.

   `diameter` is the database name
6. Create the `DIAMETERSHSUBSCRIPTION` database table and the primary key using the following commands:

   ```
   create table DIAMETERSHSUBSCRIPTION ( SERVERNAME VARCHAR2(250) NOT
   NULL, PUBLICIDENTITY VARCHAR2(250) NOT NULL, DATAREFERENCE INT NOT
   NULL, SERVICEINDICATION VARCHAR2(250) NOT NULL, CALLBACKURL
   VARCHAR2(250) NOT NULL, DATESUBSCRIBED DATE, USERID VARCHAR2(250),
   PASSWORD VARCHAR2(250), SUBSCRIBEUSERDATA SMALLINT NOT NULL ) PCTFREE
   0 PCTUSED 0 LOGGING;
   ```

   ```
   ALTER TABLE DIAMETERSHSUBSCRIPTION ADD CONSTRAINT
   PK_DIAMETERSHSUBSCRIPTION PRIMARY KEY (PUBLICIDENTITY, DATAREFERENCE,
   CALLBACKURL, SUBSCRIBEUSERDATA, SERVICEINDICATION, SERVERNAME);
   ```
7. Create the `DIAMETERROSUBSCRIPTION` database table and the primary key using the following commands:

   ```
   create table DIAMETERROSUBSCRIPTION ( SESSIONID VARCHAR2(250) NOT
   NULL, DESTINATIONREALM VARCHAR2(250) NOT NULL,  CALLBACKURI
   VARCHAR2(250), DATESUBSCRIBED DATE,  DATEEXPIRESSECONDS NUMBER(19 ,
   0) NOT NULL, USERID VARCHAR2(250), PASSWORD VARCHAR2(250) ) LOGGING;
   ```

   ```
   ALTER TABLE DIAMETERROSUBSCRIPTION ADD CONSTRAINT
   PK_DIAMETERROSUBSCRIPTION PRIMARY KEY (SESSIONID, DESTINATIONREALM);
   ```

What to do next

To improve performance, you may need to create a separate table space for the DIAMETERSHSUBSCRIPTION and DIAMETERROSUBSCRIPTION tables and complete additional database tuning.

Installing Diameter Enabler base (Cluster)

Diameter Enabler base is installed into the `plugins` directory in the WebSphere Application Server path on each federated node. The installation process includes creating and configuring a template application server for the cluster.

Before you begin

Before you begin, the following software should be installed:

- 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.
- If you require subscribe and notify functions for either Ro or Sh, one of the following databases for storing subscriber notification data:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2

In addition, you should have:

- Completed the environment preparation steps, which includes unpacking the `DHAImsConnectorInstallPackage_6.2.0.tar` file into the *`was_root`* 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`
- Prepared the database

You will need the following files for this installation task:

- `com.ibm.ws.diameter_6.2.0.jar` (in *`was_root`*`/plugins`)
- `DiameterChannelInstall.py` (in *`was_root`*`/installableApps/ImsConnector/install`)

You will need the following information to complete this task:

- WebSphere Application Server user name and password. You only need this information if security is enabled.
- Cell name and node name that will contain the first cluster member.
- Fully qualified host name that will contain the first cluster member.
- Port number for the Diameter inbound channel. Port 3868 is the preferred port defined in RFC 3588. You should use this port unless it is already in use.

Tip:

- To locate your host name and node name, refer to the `AboutThisProfile.txt` in the following node profile directory:

  AIX *`was_profile_root`*`/logs/`

  Linux *`was_profile_root`*`/logs/`

  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`*

For example, Custom01 is the name of the federated node's profile which contains the application server diameter0, created below.

- To locate your cell name, refer to the directory name found in the following location:

  AIX *was_profile_root*/config/cells/*cell_name*

  Linux *was_profile_root*/config/cells/*cell_name*

  Where:

  cell_name is your actual cell name

Important: The combination of port number and IP address must be unique for each cluster member where you plan to install Diameter Enabler base. For horizontal clusters, the same port number should be used as the IP address will be different for each server in the cluster. For vertical clusters with two or more cluster members configured to run on a single physical server with a single IP address, one cluster member can use the default Diameter port number of 3868. The additional cluster members must use different port numbers, such as 3890 or 3891. The IP address is identified as the HostIpAddress in the Web services application properties file, such as `Diameter_Sh.properties`.

About this task

Complete the following steps to install Diameter Enabler base in a clustered environment:

1. Add the Diameter Enabler base files to the deployment manager and to each federated node. Complete the following steps:

   Note: If these are on separate machines, repeat these steps for each machine.

   a. Log in to the server where the deployment manager or federated node is installed.

   b. If you have not already done so, unpack the file `DHAImsConnectorInstallPackage_6.2.0.tar` as described in preparing the environment. Otherwise, open a command prompt and verify that the file`com.ibm.ws.diameter_6.2.0.jar` was correctly copied to the`plugins` directory:

      AIX Linux *was_root*/plugins

2. Start the deployment manager.
3. Start the node agent for each federated node.
4. 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**.

5. Create an application server named diameter0. This application server will serve as the first cluster member and as a template for creating additional cluster members.
   a. In the navigation panel, click **Servers** → **Application servers**.
   b. Click **New**.
   c. Select the federated node where you want the new server created.
   d. Type `diameter0` for the **Server name**, and click **Next**.
   e. Select **default** for the server template, and click **Next**.
   f. Select **Generate Unique Ports**, and click **Next**.
   g. Click **Finish**.
   h. Click **Save** to save changes to the master configuration. You should see the following message: The configuration synchronization complete for cell.
   i. Click **OK**.

6. Configure the Diameter channels. Run the following command from the deployment manager profile directory:

   Important: Enter the following parameters on a single line.

   AIX Linux

   ```
   was_profile_root/bin/wsadmin.sh -username user_name -password
   password -f script_path/DiameterChannelInstall.py cell_name node_name
   diameter0 host_name port_number cluster [debug]
   ```

   Where:

   `user_name` represents your WebSphere Application Server user ID. This parameter is required if security is enabled.

   `password` represents the password associated with your *`user_name`*. This parameter is required if security is enabled.

   `script_path` represents the path to `DiameterChannelInstall.py`. After the unpack operation, *script_path* is *`was_root`*`/installableApps/ImsConnector/install`.

   `cell_name` represents the name of cell where diameter0 is installed

   `node_name` represents the name of node where diameter0 is installed

   `diameter0` indicates the name of the application server

   `host_name` represents the fully qualified host name the Diameter TCP channel will bind to. To specify any host name, type `"*"` (including the quotes) for the value.

   `port_name` represents the port number for the Diameter inbound TCP channel, 3868 preferred

   `cluster` indicates that the script is running in a clustered environment

   `debug` enables debugging for the configuration script (optional)

Verifying the Diameter Enabler base installation (Cluster)

Once the Diameter Enabler base is installed into the `plugins` directory, you should verify the installation.

Before you begin

Before you can verify the installation, you must have completed the installation process.

About this task

Complete the following steps to verify the installation of the Diameter Enabler base:

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. Click **Servers** → **Application servers**.
3. Click **diameter0**.
4. Under Communications, click **Ports**.
5. Verify the end point is properly configured:
   1. Verify the **DiameterNamedEndPoint** appears in the list and has the following values:

      Host: * or *host_name*

      Port: *3868* (This is the port number you defined for the Diameter inbound channel.)
6. Verify the DiameterChain is properly configured:
   1. Click **View associated transports** on the **DiameterNamedEndPoint** row.
   2. Verify **DiameterChain** appears in the list and has the following values:

      Enabled: Enabled

      Host: * or *host_name*

      Port: *3868* (This is the port number you defined for the Diameter inbound channel.)

      SSL Enabled: Disabled
7. Verify the **SecureDiameterChain** appears in the list and is properly configured with the following values:

   Enabled: Enabled

   Host: * or *host_name*

   Port: *3868* (This is the port number you defined for the Diameter inbound channel.)

   SSL Enabled: Enabled
8. Verify the DiameterChain channel is properly configured:
   1. Click **DiameterChain**.
   2. Verify the following information is correct in the **Transport Channels** section:
      - TCP inbound channel (DiameterTCPInboundChannel)
        - Host: * or *server_name*

    - Port: *3868* (This is the port number you defined for the Diameter inbound channel.)
    - Thread pool: DiameterThreadPool
  - Generic inbound channel (DiameterGenericInboundChannel)
c. Click **Generic inbound channel (DiameterGenericInboundChannel)**.
d. Verify the following values are correct:

   Transport Channel Name: DiameterGenericInboundChannel

   Discrimination weight: 10

   JAR file: com.ibm.ws.diameter_6.2.0.jar

   Channel type identifier: DiameterInboundChannel

   Configuration URI should be empty.
e. Click **Cancel** on the DiameterChain panel, and click **Cancel** to return to the Transport chain panel.

9. Verify the SecureDiameterChain channel is properly configured:
   a. Click **SecureDiameterChain**.
   b. Verify the following information is correct in the **Transport Channels** section:
      - TCP inbound channel (DiameterTCPInboundChannel)
        - Host: * or *server_name*
        - Port: *3868* (This is the port number you defined for the Diameter inbound channel.)
        - Thread pool: DiameterThreadPool
      - SSL inbound channel (DiameterSSLInboundChannel)
        - SSL Configuration: Diameter
      - Generic inbound channel (SecureDiameterGenericInboundChannel)
   c. Click **Generic inbound channel (SecureDiameterGenericInboundChannel)**.
   d. Verify the following values are correct:

      Transport Channel Name: SecureDiameterGenericInboundChannel

      Discrimination weight: 1

      JAR file: com.ibm.ws.diameter_6.2.0.jar

      Channel type identifier: DiameterInboundChannel

      Configuration URI should be empty.
   e. Click **Cancel** on the SecureDiameterChain panel, and click **Cancel** to return to the Transport chain panel.
10. Verify the SSL configuration object has been created and configured properly:
    a. In the navigation pane, click **Security** → **SSL certificate and key management** → **SSL configurations**.
    b. Click **Diameter** in the list of SSL configurations.
    c. Under Additional Properties, click **Quality of protection (QoP) settings**.
    d. Verify the following values are correct:

       Client authentication: Required

       Protocol: SSL_TLS
    e. Click **Cancel**.

Connecting to the database

After the database and WebSphere Application Server Network Deployment are installed, you must create the data source in the application server for the Ro online charging Web service to be able to access the database for the subscribe and notification functions.To connect to the database, you must create a JAAS authentication alias for the database, create the JDBC provider, and define the data source using the Integrated Solutions Console.

The database client allows the database server and the application server to communicate. When the database is not on the same server as WebSphere Application Server Network Deployment, the JDBC JAR file (or the full database client) must be installed on each WebSphere Application Server Network Deployment server. This applies to all nodes.

Creating an authentication alias

Use an authentication alias to define authentication data used to access the database. The authentication alias is created using the Integrated Solutions Console.

Before you begin

Before you begin, the following software should be installed:

- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Started the deployment manager
- Started the node agents
- Created the database

About this task

Complete the following steps to create an authentication alias for the Diameter database:

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.

   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. Expand **Java Authentication and Authorization Service**, and click **J2C authentication data**.
4. Click **New**.
5. In the **Alias** field, type `diameter_alias`. Note of the name of the alias; you will need it later.
6. In the **User ID** field, type the database administrator user ID that can be used to access the Diameter database. This is the same user ID used in the *Preparing the database* topics.
7. In the **Password** field, type the password that corresponds to the user ID.
8. Optional: In the **Description** field, type a description for the alias.
9. Click **OK**.
10. Click **Save** to save changes to the master configuration.

Creating a JDBC provider (Cluster)

To access a database, you must first create a JDBC provider in the Integrated Solutions Console.

Before you begin

Before you begin, the following software should be installed:
- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, the following steps should be completed:
- Started the deployment manager
- Started the node agents
- Created the database
- Created an authentication alias for the database

About this task

Complete the following steps to create a JDBC provider:
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 **Resources** → **JDBC** → **JDBC Providers**.
3. Select *cell_name* from the Scope drop-down list.
4. Click **New**.
5. Create the JDBC provider:
   a. Select your database type from the drop-down list.

      DB2 DB2

      Oracle Oracle

   b. Select the **Provider type** for your database:

      DB2 DB2 Universal JDBC Driver Provider

      Oracle Oracle JDBC Driver

   c. Select **Connection pool data source** for the **Implementation type**.

   d. Click **Next**.
6. For database client JAR files, perform one of the following:
   - If you have database client JAR files installed in the same location on all of the federated nodes in your cell, enter that directory location path, which is saved as a WebSphere variable.
   - If you do not have database client JAR files in the same directory location on all nodes in your cell, leave this field blank.

     Note:
       – By leaving this field blank, you will need to create or modify the WebSphere variables at the node level to represent the correct path for database client JAR files for each node.
       – In the case where the location is given here, then if it ever requires updating, make sure that the DB2UNIVERSAL_JDBC_DRIVER_PATH or ORACLE_JDBC_DRIVER_PATH variable is updated at both the cell and each node scope from **Environment** → **WebSphere Variables** in the Integrated Solutions Console.

   Then click **Next**.
7. Verify that all values are correct, and click **Finish**.
8. Click **Save** to save changes to the master configuration.
9. Click **OK** when node synchronization has completed.

Creating WebSphere variables

WebSphere variables define a parameter for the system. To enable communication between the application server and the database, you must create WebSphere variables to specify the path to the JAR files on the database client.

Before you begin

Note: In the previous topic, *Creating a JDBC provider*, you may have already entered the directory location path to these files because they are in the same location on all of the federated nodes in your cell. If the directory location path was entered, the variable has already been created; skip this procedure and proceed to the following topic, *Defining the data source*. In the future, updates to the directory location path must be made to the variable from both the cell scope and from each node's scope.

Before you begin, the following software should be installed:
- 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.
- One of the following supported databases:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- JDBC JAR file or full database client for communicating with the database server

Before you begin, the following steps should be completed:
- Started the application server
- Created the database
- Created JDBC provider

About this task

After you have installed the database client on the server where you installed WebSphere Application Server, you must create a WebSphere variable to enable communication between the application server and the database server.

Important: Install the database client in the same location on each physical server.

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** → **WebSphere Variables**.
3. Select *node_name* from the Scope drop-down list.
4. Create the JDBC driver path variable:
   a. Perform one of the following:
      - If the variable is not defined, click **New**.
      - If the variable is already defined, click its name.
   b. In the **Name** field, type:

      DB2 `DB2UNIVERSAL_JDBC_DRIVER_PATH`

        Oracle `ORACLE_JDBC_DRIVER_PATH`

   c. In the **Value** field, type the path to the database client JAR files. For the path, use the following format where *`db_client_root`* is the path where you installed the database client:

      DB2 `/`*`db_client_root`*`/java`

      Oracle `/`*`db_client_root`*`/jdbc/lib`

      This path is where you would find the following JAR files:

      DB2 db2jcc.jar and db2jcc_license_*.jar

      Oracle classes*.jar and ojdbc14*.jar

   d. Click **OK**.

5. Repeat steps 3 and 4 for each node.
6. Click **Save** to save changes to the master configuration.

Defining the data source

Data sources are the resources that provide connections to your relational database. Use the Integrated Solutions Console to define data sources.

Before you begin

Before you begin, the following software should be installed:

- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:
  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Started the deployment manager
- Started the node agents
- Created the database
- Created WebSphere variables
- Created an authentication alias for the database
- Created the JDBC provider

About this task

Complete the following steps to define the data source to map the connection to the Diameter database:

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 **Resources** → **JDBC** → **JDBC Providers**.
3. Click the name of the JDBC provider for which you are configuring the data source, to display its properties. The JDBC provider was created in the previous task.
4. Under Additional Properties, click **Data sources**.
5. Click **New**.
6. Type `Diameter` in the **Data source name** field.
7. Type `jdbc/diameter` in the **JNDI name** field.
8. Select ***diameter_alias*** from the **Component-managed authentication alias** drop-down list, and click **Next**.
9. Oracle Configure the database properties:
   a. Type `jdbc:oracle:thin:@`*`host_name`*`:`*`port_number`*`:`*`database_sid`* in the **URL** field. Where:

      host_name represents the host name of the server where Oracle Database is installed

      port_number represents the port number used to access the database server (1521 is the default port)

      database_sid represents the system identifier for the database you created

   b. Select **Oracle10g data store helper** from the **Data store helper class name** drop-down list.
   c. Click **Next**.
10. DB2 Configure the database properties:
    a. Type *`database_name`* in the **Database name** field. This is the name of the database you created.
    b. Click **4** in the **Driver type** drop-down list to specify the connectivity type of the data source. This value corresponds with the driver type property in the data source class.
    c. Type *`server_name`* in the **Server name** field. This is the fully qualified host name for the DB2 server.
    d. Type *`port_number`* in the **Port number** field. This is the port the DB2 server is listening on. (Port 50000 is the default port.)
    e. Click **Next**.
11. Select the **Use this data source in container managed persistence (CMP)** check box. Click **Next**.
12. Verify that the values are correct, and click **Finish**.
13. Click **Save** to save changes to the master configuration.
14. Click **OK** when node synchronization has completed.
15. Restart the deployment manager and the node agents.
    a. 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
```

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*.

b. Stop the node agent on each federated node. 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
```

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. Start the deployment manager. Run the following command:

AIX
```
was_profile_root/bin/startManager.sh
```

Linux
```
was_profile_root/bin/startManager.sh
```

Where:

The *was_profile_root* path contains the name of the deployment manager profile (for example, Dmgr01).

d. Start the node agent on each federated node. Run the following command:

AIX
```
was_profile_root/bin/startNode.sh
```

Linux
```
was_profile_root/bin/startNode.sh
```

Where:

The *was_profile_root* path contains the name of a federated node profile (for example, Custom01).

Testing the database connection

After creating the data source, use the Integrated Solutions Console to test the connection to the database.

Before you begin

Note: This procedure should only be used if the database client or JDBC JAR files are present, which may not be the case on the deployment manager.

Before you begin, the following software should be installed:

- WebSphere Application Server Network Deployment, version 7.0.0.1
- One of the following supported databases:

  - IBM DB2 Enterprise Server Edition, version 9.5 FixPak 1
  - Oracle Database, version 10.2.0.4, 10.2.0.6, or 11.1.0.7
- JDBC JAR file or full database client for communicating with the database server
- Diameter Enabler base

Before you begin, you should have already completed the following steps:
- Started the application server
- Created the database
- Created WebSphere variables
- Created an authentication alias for the database
- Created the JDBC provider
- Defined the data source

About this task

Complete the following steps to test the mapped database connection:

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 **Resources** → **JDBC** → **JDBC Providers**.
3. Click the name of the JDBC provider for the component you are installing.
4. Under Additional Properties, click **Data sources**.
5. Select the associated check box for the data source to test the connection.
6. Click **Test connection**. A message similar to the following indicates a successful connection:

   ```
   Test connection for data source datasource_name on server server_name at node node_name was succe
   ```

   Where:

   datasource_name represents the name of the data source you created

   server_name represents the name of the server where you created the data source

   node_name represents the node that contains *server_name*

Creating the cluster

To create the cluster, you convert an existing application server to become the first cluster member and generate additional cluster members using the original cluster member as a template.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following databases for storing subscriber notification data:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

In addition, you should have:

- Completed the environment preparation steps
- Prepared the database
- Created an application server named `diameter0`

About this task

Create the first cluster member from the diameter0 application server you have already created. Additional cluster members are added using 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 panel, click **Servers** → **Clusters**.
3. Click **New**.
4. Type `Diameter` for the **Cluster name**.
5. Select **Prefer local**.
6. Click **Next**.
7. Click **Create the member by converting an existing application server**, and select **diameter0** from the drop-down list.
8. Click **Next**.
9. Add additional cluster members. Repeat the following steps for each cluster member you would like to add.
   a. Type *`cluster_member_name`*. For federated nodes with multiple cluster members, you will need to know the names of each of the cluster members. Name each additional cluster member in the convention of `diameter+n`, such as diameter1 and diameter2.

   b. Select *node_name* from the **Select node** drop-down list. The *node_name* is the node you want the cluster to reside on.
   c. Select **Generate unique HTTP ports**.
   d. Click **Add member** to add the cluster member.

10. Click **Next**, and click **Finish**.
11. Click **Save** to save changes to master configuration. You should see the following messages:

    ADMS0200I: The configuration synchronization started for cell.

    ADMS0208I: The configuration synchronization complete for cell.
12. Click **OK**.
13. Verify the HTTP transport ports are properly defined.
    a. In the navigation panel, click **Servers** → **Application servers**.
    b. Click ***server_name*** for the cluster member you want to verify.
    c. Expand **Ports**.
    d. Verify the ports listed are correct. For vertical clusters, WebSphere Application Server automatically increments the port numbers, including the DiameterNamedEndPoint, to ensure they remain unique.

What to do next

You must create a proxy server and associate the proxy server with the cluster that you have created. Refer to the WebSphere Application Server Information Center for additional information on setting up the proxy server.

Installing the Rf accounting Web service

Rf accounting Web service is a messaging interface to enable an application to send accounting messages to a Charging Collection Function (CCF).

Installing Rf accounting Web service (Cluster)

Rf accounting Web service is installed as an enterprise application on WebSphere Application Server Network Deployment.

Before you begin

Before you begin, the following software should be installed:
- 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.
- Diameter Enabler base

Verify that you have the following file needed for the installation:
- DHADiameterRfWebServiceEAR (in *`was_root`*`/installableApps/ImsConnector`)

  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`

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. Install DHADiameterRfWebServiceEAR by completing the following steps:
   a. In the navigation panel, click **Applications** → **Install new application**.
   b. Click **Browse** to locate DHADiameterRfWebServiceEAR.
   c. Select **Show me all installation options and parameters**.
   d. Click **Next** → **Next** → **Continue**.
   e. Click **Step 2: Map modules to servers**, and select all modules.
   f. Click `WebSphere:cell=`*`cell_name`*`,cluster=`*`Diameter`* to select the cluster. If you have a Web server, also, click `WebSphere:cell=`*`cell_name`*`,node=`*`node_name`*`,server=`*`web_server`*.
   g. Click **Apply**.
   h. If security is enabled, click **Step 7: Map security roles to users or groups**, and select the check box corresponding to **Diameter_Role** to map the user and groups that you wish to grant access to the Rf accounting Web service.
   i. Click **Step 8: Summary**, and verify all options have the correct values.
   j. Click **Finish**.
   k. Click **Save** to save changes to the master configuration.

Configuring Rf accounting Web service

You need to configure Rf accounting Web service to work in your environment.

Before you begin

Before you begin, the following software should be installed:

- 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.
- Diameter Enabler base
- Rf accounting Web service

Verify that you have the following file needed:

- `Diameter_Rf.properties` (in *`was_root`*`/installableApps/ImsConnector/properties`)

  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`

About this task

You must complete the following steps on each federated node that belongs to the cluster.

1. Log in to the server where WebSphere Application Server is installed.
2. Copy `Diameter_Rf.properties` to the following directory:

   AIX Linux *`was_profile_root`*`/properties`

   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`*

   For example, Custom01.
3. Optional: If you are using vertical clustering, copy `Diameter_Rf.properties`, and rename the file based on the cluster member name. You must have a separate `Diameter_Rf.properties` for each cluster member. Each file must have a unique name and unique information within the file. Naming the file based on the cluster member name associates the configuration file with the cluster member. For example, for a cluster member named diameter0, name the configuration file `Diameter_Rf.properties.diameter0`. For a cluster member named diameter1, name the configuration file `Diameter_Rf.properties.diameter1`.
4. Open `Diameter_Rf.properties` in a text editor.
5. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
6. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
7. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.
8. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

   The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.

Installing the Ro online charging Web service

Ro online charging Web service is a messaging interface to enable an application to send online charging messages to an Online Charging System (OCS).

Note: Before installing the Web service, make sure you have completed the steps for connecting to the database.

Installing Ro online charging Web service (Cluster)

Ro online charging Web service is installed as an enterprise application on WebSphere Application Server Network Deployment.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following supported databases:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Completed the environment preparation steps.
- Started the application server.
- Connected to the database.

Verify that you have the following file needed for the installation:

- DHADiameterRoWebServiceEAR (in *`was_root`*`/installableApps/ImsConnector`)

  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`

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. Install DHADiameterRoWebServiceEAR by completing the following steps:
   a. In the navigation panel, click **Applications** → **Install new application**.
   b. Click **Browse** to locate DHADiameterRoWebServiceEAR.
   c. Select **Show me all installation options and parameters**.
   d. Click **Next** → **Next** → **Continue**.

e. Click **Step 2: Map modules to servers**, and select all modules.

f. Click `WebSphere:cell=`*`cell_name`*`,cluster=`*`Diameter`* to select the cluster. If you have a Web server, also, click `WebSphere:cell=`*`cell_name`*`,node=`*`node_name`*`,server=`*`web_server`*.

g. Click **Apply**.

h. Optional: If you are using the subscribe and notification functions, click **Step 3: Provide options to perform the EJB Deploy**.
   - **Deploy EJB option - Database type**: Choose your database:

     DB2 DB2UDB_V91

     Oracle ORACLE_V10G
   - **Database schema**: Type your database user name. This is the same user name used in *Preparing the database* section; for example, if you are using the default DB2 user name, type `db2inst1`.

i. Optional: If you are using the subscribe and notification functions, click **Step 9: Map data sources for all 2.x CMP beans**, and click **Browse** to select the **Target Resource JNDI Name** for the module. The JNDI name was created in the *Defining the data source* section.

j. If security is enabled, click **Step 13: Map security roles to users or groups**, and select the check box corresponding to **Diameter_Role** to map the user and groups that you wish to grant access to the Ro online charging Web service.

k. If security is enabled, click **Step 14: Map RunAs roles to users**, and map a user that is assigned the Diameter_Role to the RunAs role. The RoNotification EJB will run as this user when a notification is received.

   Note: Any future password changes need to be updated in this enterprise application's settings as well.

l. Click **Step 15: Summary**, and verify all options have the correct values.

m. Click **Finish**.

n. Click **Save** to save changes to the master configuration. You should see the following two messages:

   ADMS0200I: The configuration synchronization started for cell.

   ADMS0208I: The configuration synchronization complete for cell.

o. Click **OK**.

Configuring Ro online charging Web service

You need to configure Ro online charging Web service to work in your environment.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following databases for storing subscriber notification data:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

- Ro online charging Web service

Verify that you have the following file needed:

- `Diameter_Ro.properties` (in *`was_root`*`/installableApps/ImsConnector/properties`)

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`

About this task

You must complete the following steps on each federated node that belongs to the cluster.

1. Log in to the server where WebSphere Application Server is installed.
2. Copy `Diameter_Ro.properties` to the following directory:

   AIX Linux *`was_profile_root`*`/properties`

   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`*

   For example, Custom01.
3. Optional: If you are using vertical clustering, copy `Diameter_Ro.properties`, and rename the file based on the cluster member name. You must have a separate `Diameter_Ro.properties` for each cluster member. Each file must have a unique name and unique information within the file. Naming the file based on the cluster member name associates the configuration file with the cluster member. For example, for a cluster member named diameter0, name the configuration file `Diameter_Ro.properties.diameter0`. For a cluster member named diameter1, name the configuration file `Diameter_Ro.properties.diameter1`.
4. Open `Diameter_Ro.properties` in a text editor.
5. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
6. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
7. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.
8. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.

Installing the Sh subscriber profile Web service

Sh subscriber profile Web service is used by an application server to download user profile data from the Home Subscriber Server (HSS).

Note: Before installing the Web service, make sure you have completed the steps for connecting to the database.

Installing Sh subscriber profile Web service (Cluster)

Sh subscriber profile Web service is installed as an enterprise application on WebSphere Application Server Network Deployment.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following supported databases:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

Before you begin, the following steps should be completed:

- Completed the environment preparation steps.
- Started the application server.
- Connected to the database.

Verify that you have the following files needed for the installation:

- DHADiameterShWebServiceEAR (in *was_root*`/installableApps/ImsConnector`)

  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`
- `ShDataType.xsd` (in *was_root*`/installableApps/ImsConnector/xsd`)

1. Copy `ShDataType.xsd` to the *was_profile_root* directory on each federated node.

   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*
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. Install DHADiameterShWebServiceEAR by completing the following steps:

   a. In the navigation panel, click **Applications** → **Install new application**.

   b. Click **Browse** to locate DHADiameterShWebServiceEAR.

   c. Select **Show me all installation options and parameters**.

   d. Click **Next** → **Next** → **Continue**.

   e. Click **Step 2: Map modules to servers**, and select all modules.

   f. Click `WebSphere:cell=`*`cell_name`*`,cluster=`*`Diameter`* to select the cluster. If you have a Web server, also, click `WebSphere:cell=`*`cell_name`*`,node=`*`node_name`*`,server=`*`web_server`*.

   g. Click **Apply**.

   h. Optional: If you are using the subscribe and notification functions, click **Step 3: Provide options to perform the EJB Deploy**.

      - **Deploy EJB option - Database type**: Choose your database:

        DB2 DB2UDB_V91

        Oracle ORACLE_V10G

      - **Database schema**: Type your database user name. This is the same user name used in *Preparing the database* section; for example, if you are using the default DB2 user name, type `db2inst1`.

   i. Optional: If you are using the subscribe and notification functions, click **Step 9: Map data sources for all 2.x CMP beans**, and click **Browse** to select the **Target Resource JNDI Name** for the module. The JNDI name was created in the *Defining the data source* section.

   j. If security is enabled, click **Step 13: Map security roles to users or groups**, and select the check box corresponding to **Diameter_Role** to map the user and groups that you wish to grant access to the Sh subscriber profile Web service.

   k. If security is enabled, click **Step 14: Map RunAs roles to users**, and map a user that is assigned the Diameter_Role to the RunAs role. The ShNotification EJB will run as this user when a notification is received.

      Note: Any future password changes need to be updated in this enterprise application's settings as well.

   l. Click **Step 15: Summary**, and verify all options have the correct values.

   m. Click **Finish**.

   n. Click **Save** to save changes to the master configuration. You should see the following two messages:

ADMS0200I: The configuration synchronization started for cell.

ADMS0208I: The configuration synchronization complete for cell.

o. Click **OK**.

Configuring Sh subscriber profile Web service

You need to configure Sh subscriber profile Web service to work in your environment.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following databases for storing subscriber notification data:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base
- Sh subscriber profile Web service

Verify that you have the following file needed:

- `Diameter_Sh.properties` (in *`was_root`*`/installableApps/ImsConnector/properties`)

  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`

About this task

You must complete the following steps on each federated node that belongs to the cluster.

1. Log in to the server where WebSphere Application Server is installed.
2. Copy `Diameter_Sh.properties` to the following directory:

   AIX Linux *`was_profile_root`*`/properties`

   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`*

   For example, Custom01.
3. Optional: If you are using vertical clustering, copy `Diameter_Sh.properties`, and rename the file based on the cluster member name. You must have a separate `Diameter_Sh.properties` for each cluster member. Each file must have a unique name and unique information within the file. Naming the file based on the cluster member name associates the configuration file with the cluster member. For example, for a cluster member named diameter0, name the

configuration file `Diameter_Sh.properties.diameter0`. For a cluster member named diameter1, name the configuration file `Diameter_Sh.properties.diameter1`.

4. Open `Diameter_Sh.properties` in a text editor.
5. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
6. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
7. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.
8. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

   The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.

Starting the cluster

After you have created the cluster, installed, and configured the applications, you must start the cluster to synchronize the changes across the cluster and make the applications active.

Before you begin

Before you begin, the following software should be installed:

- 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.
- One of the following databases for storing subscriber notification data:

  DB2 IBM DB2 Enterprise Server Edition, version 9.1 FixPak 2

  Oracle Oracle Database, version 10.2.0.2
- Diameter Enabler base

You must have completed the following tasks:

- Created the cluster
- Installed the applications
- Configured 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. In the navigation panel, click **Servers** → **Clusters**.
3. Select the check box corresponding to the **Diameter** cluster.
4. Click **Start**.
5. Verify **Status** for the Diameter cluster is started.

Installing updates

Updates and fixes to the WebSphere IMS Connector are installed using the WebSphere Application Server version 6.1.0.x update installer.

About this task

For additional information regarding the WebSphere Application Server update installer, refer to the topic Installing maintenance packages.

The installation process consists of the following steps:

1. For each WebSphere Application Server installation being updated, install the Update Installer Plugin for IBM WebSphere products for Telecom 6.2, in .pak file format.

   This plug-in defines each component to the WebSphere Application Server update installer.

   Note: This step should be performed only one time.
2. Apply any fix packs and interim fixes that are available for the Update Installer Plugin for IBM WebSphere products for Telecom 6.2

   Note: This step should be performed only one time.
3. Use the WebSphere Application Server update installer to install the packs and interim fixes that are available for the WebSphere IMS Connector component, in .pak file format.

   These updates are installed using the standard procedure. For detailed instructions about the fix installation, refer to the Readme file that is included with every fix.

Example

Here’s an example... Insert tab A into slot B.

What to do next

Now, you too can do this...

Uninstalling IBM WebSphere Diameter Enabler Component from the WebSphere Application Server

Uninstalling the IBM WebSphere Diameter Enabler Component will remove the IBM WebSphere Diameter Enabler Component and associated Web services.

About this task

You will need the following file for this uninstallation task:

- `DiameterChannelUninstall.py` (in *`was_root`*`/installableApps/ImsConnector/install`)

  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`

Complete the following steps to uninstall Diameter Enabler from the WebSphere Application Server:

1. Start the application server. In case of a clustered installation (Network Deployment environment), start the deployment manager, all node agents, and then all application servers.
2. Uninstall all of the Diameter Enabler applications (Rf, Ro, and Sh Web services) from the administrative console:
   a. Open the WebSphere Application Server administrative console.
   b. In the navigation panel, click **Applications** → **Enterprise Application**.
   c. Select all of the Diameter Web service applications (for example, **DHADiameterRfWebServiceEAR**, **DHADiameterRoWebServiceEAR**, or **DHADiameterShWebServiceEAR**).
   d. Click **Uninstall**.
   e. Click **OK**.
   f. Click **Save**.
3. Uninstall the Diameter base from the wsadmin command line.

   Important: Enter the following parameters on a single line.

   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`*

   For example, AppSrv01 in a standalone environment, or Dmgr01 in a clustered environment.

   AIX Linux

   ```
   was_profile_root/bin/wsadmin.sh -username user_name
   -password password -f script_path/DiameterChannelUninstall.py
   cell_name node_name server_name install_type
   ```

   Where:

   `user_name` represents your WebSphere Application Server user ID. This parameter is required if security is enabled.

   `password` represents the password associated with your *`user_name`*. This parameter is required if security is enabled.

`script_path` represents the path to `DiameterChannelUninstall.py`

`cell_name` represents the name of cell where the server is installed

`node_name` represents the name of node where the server is installed

`server_name` represents the name of the application server where Diameter Enabler base is installed (first cluster member in case of a clustered installation)

`install_type` represents the environment in which the script is running, use `standalone` for a standalone environment, and `cluster` for a clustered environment

For example:

```
wsadmin.sh -f ./DiameterChannelUninstall.py west107Node01Cell west107Node01 server1 standalone
```

4. For a clustered installation only, remove the cluster.

   Note: Removing the cluster deletes the cluster and any servers associated with it. Make sure you do not have applications other than Diameter Web services applications installed on the cluster.

   a. In the navigation panel, click **Servers** → **Cluster**.
   b. Select **Diameter** cluster.
   c. Click **Delete**.
   d. Click **Save**.
5. Remove the Diameter Enabler data source:
   a. In the navigation panel, click **Resources** → **JDBC** → **Data Sources**.
   b. Select **Diameter** DataSource.
   c. Click **Delete**.
   d. Click **Save**.
6. Remove the authentication definition for the Diameter Enabler data source:
   a. In the navigation panel, click **Security** → **Secure Administration, Applications, and infrastructure** → **Java Authentication and Authorization service** → **J2C Authentication Data**.
   b. Select the database alias (<nodename>/diameter_alias).
   c. Click **Delete**.
   d. Click **Save**.
7. DB2 Remove definitions for the DB2 database variables created during the installation process:
   a. In the navigation panel, click **Environment** → **WebSphere Variables**.
   b. Perform one of the following:
      - On a standalone installation or on a clustered installation which modified this variable at the *cell* level, select the cell scope and click the **DB2UNIVERSAL_JDBC_DRIVER_PATH** variable.
      - On a clustered installation which modified this variable at each *node* level, select the node scope and click the **DB2UNIVERSAL_JDBC_DRIVER_PATH** variable.
   c. Clear the value field for this variable; then, click **OK**.
   d. On a clustered installation which modified this variable at each node level, repeat b. and c. for each node.
   e. Click **Save**.
8. Oracle Remove definitions for the Oracle database variables created during the installation process:

   a. In the navigation panel, click **Environment** → **WebSphere Variables**.
   b. Perform one of the following:
      - On a standalone installation or on a clustered installation which modified this variable at the *cell* level, select the cell scope and click the **ORACLE_JDBC_DRIVER_PATH** variable.
      - On a clustered installation which modified this variable at each *node* level, select the node scope and click the **ORACLE_JDBC_DRIVER_PATH** variable.
   c. Clear the value field for this variable; then, click **OK**.
   d. On a clustered installation which modified this variable at each node level, repeat b. and c. for each node.
   e. Click **Save**.
9. Stop the application server. In case of clustered installation (Network Deployment environment), stop all application servers, all node agents and then deployment manager.
10. Remove the file `com.ibm.ws.diameter_6.2.0.jar` from the `plugins` directory:

    AIX Linux *`was_root`*`/plugins`

    Note: In case of a clustered installation, repeat this step for each node in the cluster.
11. Remove the Diameter properties files (`Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties`) from the `properties` directory:

    AIX Linux *`was_profile_root`*`/properties`

    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`*

    For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.

    Note: In case of a clustered installation, repeat this step for each node in the cluster.
12. Remove *`was_root`*`/installableApps/ImsConnector`.

    Note: In case of a clustered installation, repeat this step for each node in the cluster.
13. Remove *`was_root`*`/lib/ext/DHAIMSConnectorTAI.jar`. In case of a clustered installation, repeat this step for each node in the cluster.

    Note: For more information, refer to the Trust Association Interceptor Information Center.
14. Remove version information:
    - *`was_root`*`/properties/version/IMSConnector.component`
    - *`was_root`*`/properties/version/IMSConnector.product`
    - *`was_root`*`/properties/version/nif/componentmaps/componentmap.ImsConnector.core.xml`

Note: In case of a clustered installation, repeat this step for each node in the cluster.

15. Clear the contents of DB2 database tables whenever you reinstall the WebSphere IMS Connector. Optionally, you can also perform this task when uninstalling the WebSphere IMS Connector. See database specific references for information on clearing database tables, removing tables and removing a database.
16. Clear the contents of Oracle database tables whenever you reinstall the WebSphere IMS Connector. Optionally, you can also perform this task when uninstalling the WebSphere IMS Connector. See database specific references for information on clearing database tables, removing tables and removing a database.

Chapter 6. Configuring IBM WebSphere Diameter Enabler Component

As the network changes, the configuration for IBM WebSphere Diameter Enabler Component may need to be updated.

Configuring Diameter Enabler base

Diameter Enabler base is configured with a unique identity during the installation process. However, changes in the network may require updates to the configuration.

Before you begin

If Diameter Enabler is moved to a different server, the configuration should be updated. Also if a proxy server is added to or removed from the network, the configuration may need to be updated.

About this task

The unique identity of Diameter Enabler is defined using the `OriginHostName`, `OriginRealmName`, and `HostIpAddress` properties in the `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` files.

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open the `Diameter_Rf.properties` in a text editor. This file is located in the following directory:

   AIX Linux `was_profile_root/properties`

   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`

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.
2. Find the `OriginHostName` property. Type the host name to match the host name of the application server where Diameter Enabler base is installed. The `OriginHostName` must be unique between a Diameter node and all of its peers. Typically, the fully qualified domain name of WebSphere Application Server where Diameter Enabler is installed can be used. However, if you are using vertical clustering, then you must alter this name for each server on that cluster. This value does not have to be identical to the fully qualified domain name of WebSphere Application Server.
3. Find the `OriginRealmName` property. Type the realm name to match the realm of the application server where Diameter Enabler base is installed.
4. Find the `HostIpAddress` property. Type the IP address where the Diameter Enabler base is installed. If Diameter Enabler base is installed on a multi-homed machine, the `HostIpAddress` should be one of the network interfaces that is present on WebSphere Application Server.

5. Find the `ProxySupport` property.
   - Type `true` to enable proxy support
   - Type `false` to turn off proxy support.

   The default value is true. If you are not using proxy servers in your environment, set the value to false. Because the proxy support settings are independent for each Web service, you can enable proxy support for one or more of the Web services.
6. Save the file and close it.
7. Open the Diameter_Ro.properties in a text editor. The file is located in the following directory:

   AIX Linux *was_profile_root*/`properties`
8. Repeat steps 2 on page 89 through 6.
9. Open the Diameter_Sh.properties in a text editor. The file is located in the following directory:

   AIX Linux *was_profile_root*/`properties`
10. Repeat steps 2 on page 89 through 6.
11. Restart the Rf, Ro, and Sh applications to make the configuration changes take affect.
    a. Log in to the Integrated Solutions Console.
    b. Navigate to the Enterprise applications page. **Click Applications** → **Enterprise applications** in the console navigation tree.
    c. Select the check box for the **DHADiameterRfWebServiceEAR**, the **DHADiameterRoWebServiceEAR**, and the **DHADiameterShWebServiceEAR** applications.
    d. Click **Stop**.
    e. Click **Start** to restart the application.

Configuring listener ports for IBM WebSphere Diameter Enabler Component

Listener ports receive incoming connections. Use the Integrated Solutions Console to view or modify the configuration of the listener ports.

About this task

You will configure listener ports during the installation process. However, as the network configuration changes, you may need to modify the listener port configuration. To configure the properties of a listener port, use the Integrated Solutions Console.

Display the collection list of listener ports.
1. In the navigation, select **Servers** → **Application Servers**.
2. In the content pane, click the name of the application server.
3. Under Communications, click **Ports**.
4. Click **DiameterNamedEndPoint**.
5. Change the properties for your listener port as needed.
6. Click **OK**.
7. Click **Save** to save changes to the master configuration.

8. Stop and restart the application server to make a changed configuration effective.

Configuring connections and routes

Connections and routes to Diameter peers must be configured in the `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` files.

About this task

Connections and routes are initially configured when you install and set up Diameter Enabler. You can configure additional connections and routes as needed. Independently configure each connection by substituting the connection number for the `x` in `conx`. Configure the remotePeerOriginHostName for your first connection using `con1.remotePeerOriginHostName=name.domain.com`. All other properties for that connection begin with `con1`. Use `con2` for the next connection, and continue numbering the connections sequentially. Add routes to the routing table for each of Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service. A route ties the final destination (realm name) to the specific connection where Diameter Enabler will send the Diameter packet.

Important:

> RFC 3588 defines a maximum of one connection between any two Diameter peers. Therefore, if you are using Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service to access a peer, they must share a single connection. For shared connections, the connection configuration information must be identical in each of the properties files that plans to use the connection.

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` in a text editor. The files can be found in the following location:

   AIX Linux *`was_profile_root`*`/properties/`

   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`*

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.
2. Modify the value of *conx.remotePeerOriginHostName* to change the name of the peer that you will be setting your TCP connection with. The `remotePeerOriginHostName` is required for the connection *conx*, and must be a fully qualified domain name.
3. Optional: Modify the value of *conx.remotePeerIpAddress* to set the IP address of the peer. If you do not configure the remotePeerIpAddress, Diameter Enabler base will perform a name resolution on the remotePeerOriginHostName. If Diameter Enabler base performs the resolution successfully, it will use the resolved IP address to set up the connection.

4. Optional: Modify the value of *conx.remotePeerPort* to change the TCP port number of the peer. The default value is 3868. If specified, Diameter Enabler base will set up the TCP connection using this port number.
5. Optional: Modify the value of *conx.inbandSecurityPolicy* to change whether or not TLS is required for all transactions. Valid settings are 0 or 1. The value 0 = PROHIBIT_TLS will not allow Diameter Enabler base to use TLS for this connection. The value 1 = REQUIRE_TLS requires Diameter Enabler base to use TLS for this connection. The default value is `0` (PROHIBIT_TLS).
6. Optional: Modify the value of *conx.watchDogTimeout* to change the number of seconds a connection can be inactive before Diameter Enabler base sends a watchdog packet to the peer. Valid values are 0 and any whole integer between 6 and 2147483647. The default value is 30. If you set the value to 0, Diameter Enabler base does not send a watchdog packet to the peer.
7. Optional: Modify the value of *conx.maxWatchDogExpirations* to change the number of watchdog timeouts after which Diameter Enabler base declares the connection suspect and starts failover processing to the secondary connection. The default value is 2. The valid values are 0, and any value between 2 to 2147483647. If you set this value to 0, Diameter Enabler base will suspend the failure monitoring, and it will not start failover processing on this connection.

   Example: For a setting of 2, when the watchdog timeout occurs, Diameter Enabler will send the first watchdog request. When the second watchdog timeout occurs, this connection and the associated routes will start failover processing.
8. Optional: Modify the value of *conx.includeOriginStateId* to indicate whether or not Diameter Enabler should transmit the Origin-State-Id AVP in Diameter base packets (CER/CEA) for this connection. A setting of true indicates Diameter Enabler should transmit the Origin-State-Id AVP in Diameter base packets (CER/CEA). A setting of false indicates the WebSphere Diameter Enabler should not transmit the Origin-State-Id AVP in Diameter base packets (CER/CEA). The default value is false.
9. Optional: Modify the value of *conx.includeFirmwareRevision* to indicate whether or not Diameter Enabler should transmit the Firmware-Revision AVP in Diameter base packets (CER/CEA) for this connection. A setting of true indicates Diameter Enabler should transmit the Firmware-Revision AVP in Diameter base packets (CER/CEA). A setting of false indicates the WebSphere Diameter Enabler should not transmit the Firmware-Revision AVP in Diameter base packets (CER/CEA). The default value is false.
10. Optional: Modify the value of *conx.maxPendingQueueLength* to change the number of requests the pending queue stores before rejecting new requests. The default setting is 30.

    Each request, when processed by the Diameter Enabler, is stored to the Pending Queue until the response to that request is received back or the request times out. So, if you are designing a system that requires a large number of requests, you may want to increase this value to a large number such as 15000. However, large queues will take up a greater amount of memory depending on the sizes of the messages held in these queues. Consequently, fine-tuning may be required to balance the trade-off in memory usage versus the number of packet transactions that can be in progress at any one time.
11. Optional: Modify the value of *conx.sourcePort* to change the source port that is used when initiating a connection to a Diameter peer. The default value is 0.

    Valid values are any integer from 1 to 65535 that is not already in use. A value of zero (0) is valid, but configures the connection to be set up with an ephemeral source port chosen by the operating system. An ephemeral source port changes every time the connection is brought up.

12. Optional: Modify the value of *conx.reconnectInterval* to change the number of seconds a connection will wait before attempting to reconnect with a peer. The default setting is 30. Valid values are any integer between 30 and 2147483647.
13. Optional: Modify the value of *conx.packetTimeout* to indicate the number of seconds that a request packet will remain on the pending queue waiting for a response packet before Diameter Enabler removes it and notifies the application of a timeout. The default value is 30.

    The packet timeout works in conjunction with the pending queue length to keep the system from backing up. You should set the timeout based on the expected response times from the Diameter Server that you are working with. In general, you should set this value substantially higher than the expected average response time. Packets that expire are removed from the pending queue, and an exception is thrown to the application that placed the initial request.
14. Set three values for each of a maximum 20 routes with the property named *routex*, where *x* is a number between 1 and 20. All three fields are required:
    - The first field is the realm name that the route represents.
    - The second field is the connection that the packet should go through to get to this realm.

      Note: The connection identifier must match the *conx* identifier that you used to configure the associated connection.
    - The third field is an indication that this route will be either a primary or a secondary route.
15. Save and close `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.
16. Restart Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

Example

Here is a example of one connection configuration:

```
con1.remotePeerOriginHostName = shserver.yourcompany.com
con1.remotePeerIpAddress = 1.2.3.4
con1.remotePeerPort = 3868
con1.inbandSecurityPolicy = 1
con1.watchDogTimeout = 30
con1.maxWatchDogExpirations = 2
con1.includeOriginStateId = false
con1.includeFirmwareRevision = false
con1.maxPendingQueueLength = 30
con1.sourcePort = 4444
con1.reconnectInterval = 30
con1.packetTimeout = 30
```

Here is an example of three possible route configurations:

```
route1 = DEFAULT:con1:PRIMARY
route2 = serviceprovider.example.com:con2:PRIMARY
route3 = serviceprovider.example.com:con3:SECONDARY
```

Channels and channel chains

Diameter Enabler uses WebSphere Application Server Channel Framework Architecture.

Channels are used to transport data between the network and Diameter Enabler. Channels are linked together to form a channel chain. Diameter Enabler supports two channel chain configurations, unsecure and secure.

During the installation process, Diameter Enabler creates two inbound channel chains, secure and unsecure. The unsecure channel chain (`DiameterChain`) uses TCP channels. The secure channel chain (`SecureDiameterChain`) adds a TLS channel on top of the TCP channel.

- The `DiameterChain` channel chain does not provide any channel security.
- The `SecureDiameterChain` channel chain is secure and uses TLS.

When Diameter Enabler attempts to establish a connection, the first packet successfully exchanged on a connection determines the direction of the chain. If Diameter Enabler initiates the connection, the chain is an outbound chain. All future packets exchanged on this connection will use this outbound chain. If the Diameter peer initiates the connection, it is an inbound chain. All future packets exchanged on this connection will use this inbound chain.

Security is configured on each connection using the `inbandSecurityPolicy` property in the `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties` files. If you plan to use secure connections, then you must have a secure channel chain configured.

Diameter Enabler base checks the first packet sent on an inbound chain to see if the data is encrypted. If the data is encrypted, the `SecureDiameterChain` channel chain is used. If the data is not encrypted, the `DiameterChain` is used. The packet security must match the chain security for the connection to be successfully established. If the packet does not match the chain, the connection will be closed.

Table 3. Channel chain configurations and resulting actions

Channel chain configuration	**Diameter Enabler configuration**	**Resulting action**
Secure inbound (`SecureDiameterChain`)	PROHIBIT_TLS	Connection is closed
	REQUIRE_TLS	Connection is successful
Unsecure inbound (`DiameterChain`)	PROHIBIT_TLS	Connection is successful
	REQUIRE_TLS	Connection is closed

Diameter Enabler configuration files

Use the `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` files to configure which Diameter peers Diameter Enabler will communicate with, and what characteristics the associated connections will have.

The `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` files are located in the `properties` directory:

AIX Linux *`was_profile_root`*`/properties`

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`

For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.

After making changes to these files you will need to restart the associated enterprise application using the Integrated Solutions Console.

In the following table every connection property that begins with the prefix `conx` is associated with the same connection and with naming the route that is using that connection. Replace the *x* in *conx* with the number of the connection that your are configuring. The value of *x* can any integer between 1 through 20.

Table 4. Property information for the Diameter_Rf.properties, Diameter_Ro.properties and Diameter_Sh.properties files

Property	**Value**
OriginHostName	Used to identify this Diameter node to those Diameter peers it shares connections with. • The *OriginHostName* is required and should be a fully-qualified domain name. **Note:** In a vertical cluster, you may have only one IP address and fully-qualified domain name. In this case, create a unique *OriginHostName* for each server in the vertical cluster. For example, add `diam1` to the FQDN to create `diam1.yourcompany.com` for the first instance, `diam2` to the FQDN to create `diam2.yourcompany.com` for the second instance, and so on. • The *OriginHostName* can be the same as the DNS hostname. • The *OriginHostName* must be unique for each Diameter client instance. • Only one Diameter client instance can be running on a given server instance. • Recommended value: use the same `hostname` as the DNS `hostname`. • Example: `OriginHostName = server.test.yourcompany.com`
OriginRealmName	The realm the Diameter client resides in. • The *OriginRealmName* is required and should be a fully-qualified domain name. • The *OriginRealmName* is only used as information exchanged in the Capabilities Exchange (CER/CEA) transaction. • Recommended value: use your current realm name. • Example: `OriginRealmName = yourcompany.com`

Table 4. Property information for the Diameter_Rf.properties, Diameter_Ro.properties and Diameter_Sh.properties files (continued)

Property	**Value**
HostIpAddress	The IP address of this Diameter client instance. - The *HostIpAddress* is required and the value will be verified against the valid IP addresses for this host server. - If the Diameter client is running on a multi-honed server, the *HostIpAddress* may be any of the IP addresses listed for that host. - This value is used as information exchanged in the Capabilities Exchange (CER/CEA) transaction. and is verified for correctness. - Recommended value: use your current host IP address. - Example: `HostIpAddress = 9.3.4.6`.
ProxySupport	A Boolean value that indicates whether this Diameter client will allow requests made from it to potentially be passed through a proxy agent in the Diameter network. - True: if set to true, packets may have the `P bit` set depending on the command code in use. - False: if set to false, no packets originating from the Diameter client will have the `P bit` set. - Default value: `True`. - Example: `ProxySupport = true`
conx.remotePeerOriginHostName	The Origin-Host name of the peer that Diameter will setup the TCP connection to. - The remotePeerOriginHostName is required for the connection "con*x*", and must be a fully-qualified domain name. - The remotePeerOriginHostName must match exactly with the configuration value on the peer as the Diameter client will only allow connections with a peer whose origin-host name matches this configured property. - Recommended value: use the host name of the remote server you are connecting to - Example: `con1.remotePeerOriginHostName = shserver.yourcompany.com`
conx.remotePeerIpAddress	IP address of the peer. - If the remotePeerIpAddress is not configured, a name resolution will be performed on the remotePeerOriginHostName. - If the name resolution is successful, the received IP address will be used to set up the connection. - Recommended value: use the host IP address of the remote server you are connecting to. - Example: `con1.remotePeerIpAddress = 1.2.3.4`

Table 4. Property information for the Diameter_Rf.properties, Diameter_Ro.properties and Diameter_Sh.properties files (continued)

Property	**Value**
conx.remotePeerPort	The TCP port number of the Peer. The default port number for the TCP connection can be overridden with a different port number. • Default value (port number): 3868. • Example: `con1.remotePeerPort = 3868`
conx.inbandSecurityPolicy	Identifies if TLS is required for all transactions. • Valid values are 0 or 1. 0 = PROHIBIT_TLS will not allow TLS to be used for this connection. 1 = REQUIRE_TLS requires that TLS is used for this connection. • Default value: 0. • Example: `con1.inbandSecurityPolicy = 1`
conx.watchDogTimeout	Number of seconds that can occur on an unused connection before a WatchDog packet (DWR) is sent to the peer. • Valid values are 0 and any value between 6 to 2147483647. • If set to zero, there will be no DWR sent to the peer. • Default value: 30 seconds. • Example: `con1.watchDogTimeout = 40`
conx.maxWatchDogExpirations	Number of consecutive watchdog timeouts that can occur before declaring this connection suspect and failing over to another available route. • Valid values are 0 and any value between 2 to 2147483647. • For a setting of 2 the DWR packet will be sent when the first watchDogTimeout occurs. When the second watchDogTimeout occurs this connection and the associated routes will start failover processing. • If set to zero the failure monitoring is suspended and failover will not occur on this connection. • Default value: 2 periods. • Example: `con1.maxWatchDogExpirations = 2`
conx.includeOriginStateId	Indicates whether or not the Origin-State-Id AVP should be transmitted in Base packets (CER/CEA). • True: indicates the Origin-State-Id AVP should be transmitted. • False: indicates the Origin-State-Id AVP should not be transmitted. • Default value: false. • Example: `con1.includeOriginStateId = false`.

Table 4. Property information for the Diameter_Rf.properties, Diameter_Ro.properties and Diameter_Sh.properties files (continued)

Property	**Value**
conx.includeFirmwareRevision	Indicates whether or not the Firmware-Revision AVP should be transmitted in Base packets (CER/CEA). • True: indicates that the Firmware-Revision AVP information will be sent. • False: indicates that the Firmware-Revision AVP information will not be sent. • Default value: false. • Example: `con1.includeFirmwareRevision = false`
conx.maxPendingQueueLength	Maximum number of requests to be stored on the pending queue. If the pending queue is full, any new requests will be rejected until room becomes available on the queue. • Default value: 30. • Example: `con1.maxPendingQueueLength = 100`
conx.sourcePort	Source port used when initiating a connection to the Diameter peer. If zero (0) is specified, an ephemeral source port is used. • Valid values are zero (0) and any integer from 1 to 65535 that is not already in use. • Default value: 0. • Example: `con1.sourcePort = 4444`
conx.reconnectInterval	Number of seconds the connection will wait before attempting to reconnect with a peer. • Valid values are any integer between 30 and 2147483647. • Default value (seconds): 30. • Example: `con1.reconnectInterval = 30`
conx.packetTimeout	Number of seconds that a request packet will remain on the pending queue waiting for a response packet before it is removed and the application is notified of a timeout. • Default value (seconds): 30. • Example: `con1.packetTimeout = 30`.

Table 4. Property information for the Diameter_Rf.properties, Diameter_Ro.properties and Diameter_Sh.properties files (continued)

Property	**Value**

routex	Defines routes used in the configuration file to link a realm to the next-hop connection.
	• Each route property value contains three fields separated by a colon (:). All three fields are required for a valid route.
	• For each realm there should be one primary route. There can be any number of secondary routes up to the maximum total of 20. The maximum number of primary routes is 10.
	• The first field is the realm name this route represents.
	• The second field is the connection that the packet to be sent to this realm should go through. The connection identifier must match the *conx* identifier that you used to configure the associated connection.
	• The third field is an indication this route will be either a primary or a secondary route. A secondary route will only be used for forwarding traffic if no primary route can be reached.
	• Example:
	`route1 = DEFAULT:con1:PRIMARY`
	`route2 = example.com:con2:PRIMARY`
	`route3 = example.com:con3:SECONDARY`

subscriptionPurgingInterval	This property only applies to Ro. In a clustered installation, this property should only be enabled on a single application server. Defines the interval in which expired subscriptions to receive Reauthorization Requests are purged from the database. The minimum value is 30 seconds, and the maximum value is 32,000,000 seconds. If the property is missing or commented out (default), purging will not occur. • Default value: Purging is suspended • Example: `subscriptionPurgingInterval = 30`.

Chapter 7. Securing IBM WebSphere Diameter Enabler Component

IBM WebSphere Diameter Enabler Component utilizes the security functionality provided in WebSphere Application Server Network Deployment.

Note: The Trust Association Interceptor (TAI) security component is not intended to be used with the IBM WebSphere Diameter Enabler Component. The TAI is intended for extending security trust through a virtual cloud on the front end of an application server that is servicing HTTP and SIP requests. These are HTTP and SIP requests that have passed through a Reverse Proxy Security Server (RPSS) that has performed some level of authentication on the source. The RPSS notion is not applicable to Diameter Web service transactions. Therefore, the TAI should not be configured on the same platform that you run the Diameter Enabler.

Channel security

Diameter Enabler supports IPsec or transport layer security (TLS) on each connection in the channel chain. If TLS is enabled, then the signer certificates on the WebSphere Application Server must support the inbound signer key certificate.

You may use IPsec or TLS to secure Diameter connections between the WebSphere Diameter Enabler and its peers. Implement IPsec as either the operating system configuration or an external device that encrypts and decrypts the encoded messages. In this capacity, IPsec usage is invisible to the WebSphere Diameter Enabler. If you are using IPsec, configure the Diameter connections in the `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` to `PROHIBIT_TLS`, because it is unnecessary in an IPsec environment. If you do not have IPsec, configure the connections to be secured through TLS. Otherwise, Diameter Enabler will send Diameter messages through an unsecure network.

IPsec is the default security protocol that Diameter Enabler will look for when establishing a connection. If IPsec is not present and security has been configured in the properties file, then a secure channel will be used. IPsec is implemented using either the operating system configuration, or an external device that encrypts and decrypts the encoded messages.

In order for the SecureDiameterChain to work properly, the peer and the application server must exchange signer certificates. According to RFC 3588, the Diameter nodes must mutually authenticate. This means that the client authentication parameter in the SSL configuration must be set to `Required`. Diameter Enabler nodes must support the following cipher suites:

SSL_RSA_WITH_AES_128_CBC_SHA
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA

Both ends of the connection must supply certificates so that authentication is performed in both directions. This is handled through the SSL channel configuration. If TLS is used, it must be supported by all the Diameter Enabler devices. If TLS is used, both the WebSphere Diameter Enabler and the Diameter peer that it is connecting to should be configured for TLS.

Configuring channel security

You can enable transport layer security by modifying the properties file for Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

About this task

The `inbandSecurityPolicy` property is in the `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` files. Prior to establishing a connection to a remote Diameter peer using TLS, the WebSphere Application Server and the remote Diameter peer must perform a certificate exchange. The Diameter protocol requires mutual authentication between the Diameter peers, which is a two-step process: the WebSphere Application Server Exports a signer certificate and the Diameter peer imports it into the keystore; the Diameter peer exports the signer certificate and the WebSphere Application Server imports it into the keystore. The installation process creates a new SSL configuration object called Diameter that is associated with the TLS channel that is part of the SecureDiameterChain channel chain. As a default, the SSL configuration called Diameter is set up to use the `NodeDefaultKeyStore` and `NodeDefaultTrustStore`.

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`. The files are in the following location:

   AIX Linux *`was_profile_root`*`/properties`

   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`*

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile, in a clustered environment.
2. Set the value for `inbandSecurityPolicy`.
   - To enable TLS on a specific connection, set the value for the property to `1`.
   - To disable TLS on a specific connection, set the value for the property to `0`.
3. Save and close `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.
4. Restart Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

What to do next

Note: If you enable TLS, the WebSphere Application Server must support the inbound signer key certificate.

Modifying channel security

Diameter Enabler supports the ciphers supported by the RFC 3588 specification.

About this task

The installation process creates a new SSL configuration object called Diameter that is associated with the TLS channel that is part of the SecureDiameterChain channel chain. As a default, the SSL configuration called Diameter is set up to use the `NodeDefaultKeyStore` and `NodeDefaultTrustStore`. You can modify these, but you must import the signer certificate from the Diameter peer into the keystore associated with this configuration. Additionally, you must export the default signer associated with the trust store from the trust store associated with this configuration and import it to the Diameter peer trust keystore. The installation process sets the Client authentication on the SSL configuration to **Required**. Because it is required by RFC 3588, this should not be modified.

1. Log in to the Integrated Solutions Console.
2. Click **Security** → **SSL certificate and key management** → **SSL configurations**.
3. Click **Diameter**, the SSL configuration created during the installation process.
4. Click **Quality of protection (QoP) settings**.
5. Select **Required** in the **Client authentication** drop-down.
6. Modify the ciphers as needed.
7. Click **OK** to save your changes.

Chapter 8. Administering IBM WebSphere Diameter Enabler Component

Various IBM WebSphere Diameter Enabler Component usage can be monitored using the Integrated Solutions Console, while other component features can be modified by changing the appropriate properties file.

You can monitor thread usage using the Integrated Solutions Console and modify the watchdog timeout, pending queue maximum, and reconnect interval by changing values in `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.

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`*

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
```

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
```

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
```

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
```

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
```

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
```

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.

Viewing channel chains

You can view the Channel chains configured for Diameter Enabler through the administration console.

About this task

For more information on what type of security a *DiameterNamedEndPoint* is using, complete the following steps to view the associated transport information in the Integrated Solutions Console:

Note: Repeat the following steps for each server, if more than one.

1. Log in to the Integrated Solutions Console.
2. Click **Servers** → **Application Servers** → ***server_name*** → **Ports**
3. Find the entry for port name equal to *DiameterNamedEndPoint*. Click **View associated transports**.

Monitoring threads

You can monitor and log IBM WebSphere Diameter Enabler Component thread usage.

About this task

Complete the following steps:

Note: If you plan to monitor an application server that is in a cluster, make sure that both the server and its node agent are running before proceeding with 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. Click **Monitor And Tuning** → **Performance Viewer** → **Current Activity**.
3. Click ***server_name***.
4. Expand **Performance Modules**.
5. Expand **Thread Pools**
6. Select **DiameterThreadPool**
7. Click **View Modules**.
8. Click **Start Logging** to start logging performance data. Once you start monitoring for your server, you will be able to view real time operation in the Tivoli Performance Viewer panels. Click **More information about this page** for more information about logging and the Tivoli Performance Viewer.

Adjusting heap size for subscription database handling

You can adjust your heap size for subscription database handling by changing the maximum heap size in WebSphere Application Server.

About this task

The purging thread is intended to periodically remove subscription entries that have expired in the database. The proper handling is for an application to subscribe for an Ro Reauthorization Notification immediately before it starts using a session ID and then unsubscribing for that session ID once the authorization session has finished. If the application does not unsubscribe and the subscriptionDuration interval has passed, then that subscription entry is a candidate for deletion by the purging thread.

The purging thread does not run by default. If a *subscriptionPurgingInterval* is configured, the purging thread will run repeatedly with a delay of the purgingInterval in between runs. The purging thread retrieves all expired subscriptions in the subscription table and deletes them each pass. If the table is very large, and the number of expired subscriptions is expected to be very large, then the configured heap size must also be very large to handle these large reads.

While the heap size varies depending on the mix of application data and IMS Connector data that is present, the following basic guidelines will help you determine the size of heap that is required:

- Each subscription record varies in size, but a reasonable estimate is a 550 byte entry. If you expect the system to be able to purge 1,000,000 entries at a time, then you must add 550 MB of heap to your existing heap to handle this surge in entries. Likewise, if you expect to purge 2,000,000 entries at a time, you must add 1100 MB of heap space to the existing heap.
- Purging a table where more than 2,000,000 entries are expired should be handled through an independent database script where the script periodically deletes all entries that are expired through an SQL query.

Note: For information about changing the maximum heap size, see the "Java virtual machine settings" topic, located in the Reference section, in the WebSphere Application Server 7.0 Information Center.

If you wish to continue to use the purging thread for large queries, you should make the *subscriptionPurgingInterval* as small as possible (e.g., 30 seconds). This has the affect of processing the expired entries more frequently; consequently, the number of entries to be purged at a time is less. If entries expire at a faster rate than they can be removed, then the number of expired entries will grow indefinitely. The result will be that the heap space will be eventually exhausted and an out of memory condition will occur.

Modifying the watchdog timeout interval

You can modify the watchdog timeout by changing the connection timeout time, which is the time a connection can be inactive before a watchdog message is sent. You can also change the number of consecutive unanswered watchdog requests that can occur before a connection is placed on a Suspect Peer Connections List.

About this task

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` in a text editor. The files can be found in the following location:

   AIX Linux *`was_profile_root`*`/properties/`

   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`*

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.

2. Modify the value of *conx.watchDogTimeout* to change the number of seconds a connection can be inactive before Diameter Enabler base sends a watchdog packet to the peer. Valid values are 0 and any whole integer between 6 and 2147483647. The default value is 30. If you set the value to 0, Diameter Enabler base does not send a watchdog packet to the peer.
3. Modify the value of *conx.maxWatchDogExpirations* to change the number of watchdog timeouts after which Diameter Enabler base declares the connection suspect and starts failover processing to the secondary connection. The default value is 2. The valid values are 0, and any value between 2 to 2147483647. If you set this value to 0, Diameter Enabler base will suspend the failure monitoring, and it will not start failover processing on this connection.

   Example: For a setting of 2, when the watchdog timeout occurs, Diameter Enabler will send the first watchdog request. When the second watchdog timeout occurs, this connection and the associated routes will start failover processing.
4. Save and close `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.
5. Restart Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

Example

Example:

```
con1.watchDogTimeout = 40
```

```
con1.maxWatchDogExpirations = 2
```

Modifying the pending queue maximum

You can modify the maximum number of requests the pending queue stores before rejecting new requests.

About this task

Each connection in Diameter Enabler environment contains a pending queue which holds references to all of the requests on that connection. When Diameter Enabler receives a response, it will remove the corresponding request from the queue. If a packet is in the queue longer than the `packetTimeout`, Diameter Enabler will remove it from the queue and return an error to the sending Application. If the number of requests in the queue reaches this `maxPendingQueueLength` value, Diameter Enabler will reject any new requests until room is available on the queue.

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` in a text editor. The files can be found in the following location:

   AIX Linux `was_profile_root/properties/`

   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`*

For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.

2. Modify the value of *conx.maxPendingQueueLength* to change the number of requests the pending queue stores before rejecting new requests. The default setting is 30.

   Each request, when processed by the Diameter Enabler, is stored to the Pending Queue until the response to that request is received back or the request times out. So, if you are designing a system that requires a large number of requests, you may want to increase this value to a large number such as 15000. However, large queues will take up a greater amount of memory depending on the sizes of the messages held in these queues. Consequently, fine-tuning may be required to balance the trade-off in memory usage versus the number of packet transactions that can be in progress at any one time.
3. Save and close `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.
4. Restart Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

Example

Example:

```
con1.maxPendingQueueLength = 30
```

Modifying the source port

You can modify the source port used when initiating a connection to the Diameter peer.

About this task

The `conx.sourcePort` setting is used to force the source port to be a specific, fixed value when initiating a connection to the Diameter peer. This is helpful when configuring firewalls to screen traffic based on source port information.

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` in a text editor. The files can be found in the following location:

   AIX Linux `*was_profile_root*/properties/`

   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*`

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.
2. Modify the value of *conx.sourcePort* to change the source port that is used when initiating a connection to a Diameter peer. The default value is 0.

   Valid values are any integer from 1 to 65535 that is not already in use. A value of zero (0) is valid, but configures the connection to be set up with an ephemeral source port chosen by the operating system. An ephemeral source port changes every time the connection is brought up.

3. Save and close `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.
4. Restart Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

Example

Example:

```
con1.sourcePort = 4444
```

Modifying the reconnect interval

You can modify the number of seconds a connection will wait before attempting to reconnect with a peer.

About this task

Note: In case of a clustered installation, repeat these steps for each node in the cluster.

1. Open `Diameter_Rf.properties`, `Diameter_Ro.properties`, and `Diameter_Sh.properties` in a text editor. The files can be found in the following location:

   AIX Linux *`was_profile_root`*`/properties/`

   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`*

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.
2. Modify the value of *conx.reconnectInterval* to change the number of seconds a connection will wait before attempting to reconnect with a peer. The default setting is 30. Valid values are any integer between 30 and 2147483647.
3. Save and close `Diameter_Rf.properties`, `Diameter_Ro.properties`, or `Diameter_Sh.properties`.
4. Restart Rf accounting Web service, Ro online charging Web service, or Sh subscriber profile Web service.

Example

Example:

```
con1.reconnectInterval = 30
```

Modifying the maximum packet size

Modify the channel chains to regulate the maximum packet size, in bytes, that Diameter Enabler can send or receive.

About this task

The **maxPacketSize** parameter is primarily intended to protect the system from excessive resource use by a rogue or erroneous applications or Diameter peers.

You can modify the custom property **maxPacketSize** for Diameter Enabler. If a peer attempts to send a packet that is larger than the defined **maxPacketSize** value, Diameter Enabler will log an error and close the connection with the peer. If one of the Diameter applications attempts to use Diameter Enabler to send a packet that exceeds the defined **maxPacketSize**, the send request will fail, and Diameter Enabler will return an error to the application and keep the connection open. If the **maxPacketSize** value is not defined, Diameter Enabler will use a default value of 10000 bytes.

You must set the **maxPacketSize** value separately on each of the Diameter channel chains, **DiameterChain** and **SecureDiameterChain**. After you set the **maxPacketSize**, you must restart the application server for the change to take effect. Therefore you may need to schedule that update to the application server.

1. Log in to the Integrated Solutions Console.
2. Click **Servers** → **Application Servers** → *server_name* → **Ports**.
3. In the Port Name column, find the **DiameterNamedEndPoint**.
4. In the Transport Details column, click **View associated transports**.
5. Click **DiameterChain** or **SecureDiameterChain**.
6. Click **Generic inbound channel (DiameterGenericInboundChannel)**.
7. Click **Custom Properties**.
8. Click **New**.
9. Under **General Properties**, type `maxPacketSize` in the Name field.
10. Type a value for this property in the Value field. The packets are measured in bytes.
11. Optional: You can also specify a description of this property in the Description field.
12. Click **Apply** or **OK**.
13. Click **Save** to save your configuration changes.
14. Restart the application server.

Modifying the Work Manager settings

Modify the Work Manager to regulate the number of threads that handle outbound Web service notifications to application clients.

About this task

You can modify the minimum and maximum number of threads for the Diameter Enabler Work Manager. In Ro online charging Web service, this has a direct impact on the number of threads that can process incoming Reauthorization Requests at any given time. In Sh subscriber profile Web service, this has a direct impact on the number of threads that can process incoming Push-Notification-Requests.

If the rate of notifications is greater than the rate that the application server can process them with the allotted thread pool and work request queue size, the application will log an error and the notifications will fail with a WorkRejectedException. In this scenario, the minimum number and maximum number of threads can be raised to allow the processing of more simultaneous requests. Raising the number of threads will increase memory and CPU utilization. Consequently, the trade-off between resource utilization and performance should be evaluated.

After updating any Work Manager settings, you must restart the application server for the change to take effect. Therefore, you may need to schedule that update to the application server.

1. Log in to the Integrated Solutions Console.
2. Click **Resources** → **Asynchronous beans** → **Work managers** → **DiameterWorkManager**.
3. In the Minimum Number of Threads field, type the following: `min_num_threads`
4. In the Maximum Number of Threads field, type the following: `max_num_threads`
5. Click **Apply** or **OK**.
6. Click **Save** to save your configuration changes.
7. Restart the application server.

Modifying the subscription purging interval

You can modify the interval in which expired subscriptions set to receive reauthorization requests are purged from the database. This property only applies to Ro online charging Web service.

About this task

If the *subscriptionPurgingInterval* property is missing or commented out (default), purging will not occur. The default value is Purging is suspended. The minimum value is 30 seconds, and the maximum value is 32,000,000 seconds. If you are designing a system that requires a large number of subscription requests, you may want to adjust the purging interval to prevent large numbers of records from being purged at one time. If a large number of records require purging when the end of interval is reached, the performance of the overall system may be affected.

Important: In a clustered installation, this property should only be enabled on a single application server.

1. Open `Diameter_Ro.properties` in a text editor. The file can be found in the following location:

   AIX Linux *`was_profile_root`*`/properties/`

   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`*

   For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.
2. Modify the value of *subscriptionPurgingInterval* to change the interval in which expired "subscriptions to receive ReAuth Requests" are purged from the database. If the property is missing or commented out (default), purging will not occur. The default value is Purging is suspended. The minimum value is 30 seconds, and the maximum value is 32,000,000 seconds.
3. Save and close `Diameter_Ro.properties`.

4. Restart Ro online charging Web service.

Example

Example:

```
subscriptionPurgingInterval = 30
```

Modifying the user mapped to the RunAs role

You can modify the user mapped to the RunAs security role.

About this task

If security is enabled, then the Diameter Enabler Web service enterprise applications will have one or more users or groups assigned to the security role, ″Diameter_Role.″ One of these users, or members of these groups, will be mapped to the RunAs role. These settings are normally configured during the install of the enterprise applications. However, if a different user must be mapped to the RunAs role or if the existing user’s password changes, then the enterprise applications settings must be updated to match.

If the user ID password is changed in the user registry but not in the enterprise application settings, then you may see messages in `System.out` similar to the following:

- `SECJ0055E: Authentication failed for` *`user ID`*`. The user ID or password may have been entered incorrectly or misspelled. The user ID may not exist, the account could have expired or disabled.  The password may have expired.`
- `SECJ0336E: Authentication failed for user` *`user ID`* `because of the following exception com.ibm.websphere.security.PasswordCheckFailedException: Authentication failed for user:` *`user ID`*

To modify this user ID or password, perform 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. Click **Applications** → **Enterprise Application** → *application_name* → **User RunAs roles**
3. Select the **Diameter_Role** check box.
4. Enter the current RunAs user ID or a new user ID in the Username field.

5. Enter the current password for this user ID in the Password field.
6. Click **Apply**.

   Note: The user ID/password combination must be valid in the user registry that is configured and must already be assigned to the security role (Diameter_Role); if not, this step will fail.
7. Click **OK**.
8. Click **Save** to save your configuration changes.
9. Restart the application server.

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 Diameter Enabler Component 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 supported by Diameter Enabler. Review the ITLM documentation carefully to determine which operating systems, maintenance levels, and Linux kernel versions are supported.

Installation

During the installation of Diameter Enabler, the inventory signatures for ITLM are installed in the following directory:

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`

For example, AppSrv01 in a standalone environment, or Custom01 as the name of a federated node profile in a clustered environment.

`was_profile_root/installedApps/cell_name/DHADiameterXxWebServiceEAR.ear/itlm`

Where:

cell_name is your configuration cell name

Xx is either Rf, Ro or Sh

Inventory Signature

For Diameter Enabler, an ITLM inventory signature file uniquely identifies each subcomponent and is installed with the Diameter Enabler EAR into WebSphere Application Server. The file names are as follows:

Rf subcomponent: `IMSDRF0602.SYS2`

Ro subcomponent: `IMSDRO0602.SYS2`

Sh subcomponent: `IMSDSH0602.SYS2`

License file

Each Diameter Enabler subcomponent comes with a license file that is not associated with ITLM enablement, but is still important for licensing purposes. This text file specifies the customer’s entitlement of installation and use of a specific subcomponent. The licenses are located in the `License` subdirectory on the CD.

Usage Signature

The ITLM usage signature, generated by the Software Catalogue Signature team, is used to identify each component as a J2EE product.

Chapter 9. Troubleshooting IBM WebSphere Diameter Enabler Component

Logs store information to help you troubleshoot problems installing, configuring, and using IBM WebSphere Diameter Enabler Component.

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 IP Multimedia Subsystem Connector. 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.

Monitoring log messages

IBM WebSphere Diameter Enabler Component 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, IBM WebSphere Diameter Enabler Component 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*

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*

In a standalone environment, *profile_name* is the name of the application server profile (for example, AppSrv01). In a clustered environment, *profile_name* is the name of a federated node profile (for example, Custom01).

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. IBM WebSphere Diameter Enabler Component messages are also identified with a class name. For example:

```
[4/8/09 10:45:14:773 EDT] 00000031 DiameterState I DHAC0100I: Connection
has transitioned to the Open state. Peer Host: hostname.myco.com
ApplicationId[0] = VendorId = 10415; Auth Application Id = 16777217;
Supported vendor ids = 10415
```

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.
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.

Hang detection policy

The WebSphere Application Server has a hang detection policy for monitoring threads that have been active longer than the time defined by the thread monitor threshold.

Each Diameter Enabler connection is associated with a thread that remains active for the duration of the connection. Since this duration is typically much larger than the thread monitor threshold, warnings are logged in the WebSphere Application Server System.Out log file that indicate the name of the thread that is hung and how long it has already been active. The warning messages are in the following format:

```
WSVR0605W: Thread threadname has been active for hangtime and may be hung.
There are totalthreads threads in total in the server that may be hung.
```

Disabling the hang detection policy disables the monitoring of all thread pools residing in WebSphere Application Server and is not recommended. However, you can configure the hang detection policy using the procedure *Detecting hung threads in J2EE applications* in the WebSphere Application Server Information Center.

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 IBM WebSphere Diameter Enabler Component 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:
   1. Display the Runtime tab.
   2. 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**.
   3. Click **Change Log Level Details**.
   4. Click **Components** to view all loggers for the individual components.
   5. Click **+** to show the *children* of the logger.
   6. Click ***logger_name*** to change the log details. To enable tracing on specific components of IBM WebSphere Diameter Enabler Component, click one of these logger groups:
      - com.ibm.diameter.*
      - com.ibm.diameter.base.*
      - com.ibm.diameter.cfsm.*
      - com.ibm.diameter.charging.*
      - com.ibm.diameter.packet.*
      - com.ibm.diameter.rf.*
      - com.ibm.diameter.ro.*
      - com.ibm.diameter.sh.*
      - com.ibm.ws.diameter.*
   7. 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. Optional: To control logging of information about offline accounting and online charging functions, you can configure trace options for Rf accounting Web service and Ro online charging Web service usage information.
   1. Click **com.ibm.diameter.rf.usage** to change the log details. The com.ibm.diameter.rf.usage logger controls the logging of information about the beginning and ending of offline accounting functions.
   2. Click **com.ibm.diameter.ro.usage** to change the log details. The com.ibm.diameter.ro.usage logger controls the logging of information about the beginning and ending of online charging functions.
   3. Click **Message and Trace Levels** → **info** to log usage information.

   The messages will be logged in `rfUsage_x.log` and `roUsage_x.log`, where *x* is a number from 0 to 9. The maximum size of the log file is 100 Mb. When the log file reaches the maximum size, a new log will be created. After the tenth file has been created, the first file will be overwritten. For example, assume `rfUsage_0.log` (or `roUsage_0.log`) is the first file created. When `rfUsage_9.log` (or `roUsage_9.log`) reaches the maximum size, `rfUsage_0.log` (or `roUsage_0.log`) will be overwritten with the latest information. The `rfUsage_x.log` and `roUsage_x.log` files will be created in the following location:

   AIX `was_profile_root/logs/server_name`

   Linux `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`*

In a standalone environment, *profile_name* is the name of the application server profile (for example, AppSrv01). In a clustered environment, *profile_name* is the name of a federated node profile (for example, Custom01).

8. Click **OK**.
9. 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.

Note: The table lists the names of trace loggers that are in each parent log level. For example, if the table lists the trace group as `com.ibm.imsconnector.tai.*` and the trace logger as `*SipInterceptor`, then the full name of the trace logger is `com.ibm.imsconnector.tai.SipInterceptor`.

To control the trace level for the Diameter Enabler component and its subcomponents, use the options on the `com.ibm.diameter.*` trace group. To control the trace level for the Diameter channel, use the options on the `com.ibm.ws.diameter.*` trace group. For more specific levels of tracing, use the following trace groups, which are relevant to Diameter Enabler:

Table 5. Diameter trace groups and trace loggers

Trace group	**Trace loggers**
com.ibm.diameter.base.*	*DiameterBaseApiHelper *DiameterBaseApiImpl *osgi
com.ibm.diameter.cfsm.*	*DiameterCfsm *DiameterChannelReader *DiameterChannelWriter *DiameterConnection *DiameterConnectionPacketUtil *DiameterProtectedAppIdArrayList *DiameterState_I_Closed *DiameterState_I_Closing *DiameterState_I_WaitCea *DiameterState_I_WaitConnAck *DiameterState_RI_Open *DiameterState_R_Closed *DiameterThreadMgr

Table 5. Diameter trace groups and trace loggers (continued)

Trace group	Trace loggers
com.ibm.diameter.charging.*	*util
com.ibm.diameter.packet.*	*Avp *AvpParser *DiameterPacket *VsAvp
com.ibm.diameter.rf.*	*avp *base *command *usage *util
com.ibm.diameter.ro.*	*base *command *notify *notify.subscription *usage *util
com.ibm.diameter.sh.*	*avp *base *command *notify *notify.subscription *util
com.ibm.ws.diameter.*	*channel.impl.DiameterTcpConnection *channel.inbound.impl.DiameterInboundChannelFactory *channel.inbound.impl.DiameterTcpInboundChannel *channel.outbound.impl.DiameterTcpOutboundChannel *channel.outbound.impl.DiameterTcpOutboundConnLink
com.ibm.ims.*	*normalize*

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.*

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.

IBM WebSphere Diameter Enabler Component messages

The following message identifiers are used for Diameter Enabler:

Table 6. Diameter Enabler

Component identifier	**Component description**
DHAP	Diameter Enabler packet messages
DHAC	Diameter Enabler connection messages
DHAS	Sh subscriber profile Web service messages
DHAR	Rf accounting Web service messages
DHAO	Ro online charging Web service messages
DHAG	Common Rf accounting Web service and Ro online charging Web service messages

Chapter 10. Developing applications that use Diameter Web services

IBM WebSphere Diameter Enabler Component provides Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service.

Rf accounting Web service, Ro online charging Web service, and Sh subscriber profile Web service are collections of Web service methods that use SOAP over HTTP Request and Reply interface. The collections of service methods in Rf and Ro consist of a set of well-defined operations to perform offline and online session and event charging. The collection of service methods in Sh provides subscriber profile services. The Rf accounting Web service and Ro online charging Web service include raw methods to send custom-built packets. To use the raw methods, the user must have a basic knowledge of the Diameter protocol and packet structure. The raw methods require use of the IBM WebSphere Telecom Toolkit.

Diameter Web services are stateless classes that many threads can access simultaneously. In each Web service request, the required objects for processing the request and reply will be created. No internal state information will be retained in the Web service classes.

Rf accounting Web service

Rf accounting Web service provides an IMS Application Server application with a Diameter messaging interface to enable the application to send accounting messages to Charging Collection Function (CCF). The CCF builds a Charging Data Record (CDR) which is sent to and consumed by the billing system.

The IMS Application Server communicates with the accounting server through Rf accounting Web service using either a session or event offline charging method as shown here:

- Session charging - Charging for a session that takes place over a period of time.
  - Start: Starts an accounting session.
  - Interim: Periodically updates the accounting session.
  - Stop: Stops the accounting session.
- Event charging - Accounting transaction for a single operation.

The choice of using session charging versus event charging will depend on the applications using this interface. The actual accounting functionality is based on other network elements, such as billing, and is not part of the Rf accounting Web service. The Rf interface defines several external Web service methods used for offline accounting messages, start session, interim session, stop session and event charging.

Application overview

The following is a brief overview of the system components and how they play in a Diameter transaction:

- IMS Application Server Application: The IMS Application Server Application uses WSDL and/or helper classes to prepare a Web service request to be sent to one of the Diameter Enabler Web service applications.

  If the Diameter Enabler Web service supports notifications, the IMS Application Server application must use the WSDL and/or helper classes to implement the ″server″ Web service that receives notifications. The URL of the ″server″ Web service is provided to the Diameter Enabler when invoking subscription requests.
- Diameter Enabler: The Diameter Enabler is capable of receiving Web service requests and converting them into Diameter request packets. The Diameter Enabler receives the Web service request, validates its contents, converts it to a Diameter request packet, and routes it to the correct Diameter Server. The Diameter Server responds with a Diameter response packet, which is returned as a Web service response object to the IMS Application Server application that performed the request.

  The Diameter Enabler is also capable of receiving notification requests from a Diameter server. The Diameter Enabler validates the contents of the Diameter request packet and responds to the peer with a Diameter answer packet. If the IMS Application Server Application has created a subscription to receive the notifications, the Diameter Enabler opens an outbound Web service request to a Web service endpoint provided in the subscription data.
- Diameter Server: The Diameter Server is a Diameter peer that exchanges Diameter messages with the Diameter Enabler. The Diameter Server is capable of sending and receiving Diameter request and answer packets.

Application deployment

Rf accounting Web service is deployed in a WebSphere Application Server environment. Prior to installing Rf accounting Web service, you must install Diameter Enabler base. Any application with the appropriate access and security may invoke these functions while the supported transport is SOAP over HTTP. The Rf accounting Web service are stateless and support multiple simultaneous calls.

Rf accounting Web service call flows

Rf accounting Web service call flows describe how the application transmits and receives accounting packets and interacts with the application server, the Diameter Enabler base, and the CCF.

Diameter Enabler base supports multiple realms. The internal routing table controls which realm a packet is sent to.

The Rf accounting Web service routing table is independent from each of the other services (Sh and Ro). Although all of the Web services use the Diameter Enabler base, the configuration and usage of the routing table remains independent.

Rf accounting Web service uses a consistent flow for all Accounting messages:

1. An IMS Application Server invokes one of the Rf accounting Web service requests, such as startRfAccounting.
2. The Rf accounting Web service application validates the parameters received from the Web service interface and then builds an Rf Accounting Request (ACR) message using those parameters. Diameter Enabler base then sends the Accounting Request message to the CCF.

3. The CCF receives the Accounting Request message, updates the Charging Data Record, and returns the result to the Diameter Enabler base in an Accounting Answer (ACA) message.
4. The Rf accounting Web service application receives the Accounting Answer message from the Diameter Enabler base that contains the result of the Accounting operation. Then the Rf accounting Web service application returns the result to the application that invoked the Web service request.

Rf accounting Web service methods

Rf accounting Web service defines several external Web service methods for managing an offline charging session or issuing a one-time offline charging event.

According to RFC 3588 the **sessionID** is an agreed upon format between the IMS Application Server and the accounting server. The same **sessionID** is expected to be used for each request associated with a session. The session is terminated when an accounting stop message is sent using stopRfAccounting.

Web service accounting methods

The following methods can be used to send offline charging information through the Web service interface. Rf accounting Web service supports the following external methods that are available to the IMS Application Server applications. For detailed information on method signatures and descriptions, refer to the com.ibm.diameter.rf.DiameterRfService_SEI section in the Javadoc.

Table 7. Web service offline charging methods

Rf accounting method	**Description**
RfAccountingResults* startRfAccounting(RfAccountingInfo** rfAcctInfo);	Starts an accounting offline charging session
RfAccountingResults* stopRfAccounting(RfAccountingInfo** rfAcctInfo);	Stops an accounting offline charging session
RfAccountingResults* interimRfAccounting(RfAccountingInfo** rfAcctInfo);	Updates an accounting offline charging session
RfAccountingResults* eventRfAccounting(RfAccountingInfo** rfAcctInfo);	Processes a one-time offline charging event
ACAResults*** startOfflineAccounting(java.lang.String sessionId, int recordNumber, java.lang.String userName, int acctInterimInterval, java.lang.String destinationRealm, long eventTimestamp, int originStateID, Accounting act);	Starts an accounting offline charging session (deprecated)
ACAResults*** stopOfflineAccounting(java.lang.String sessionId, int recordNumber, java.lang.String userName, int acctInterimInterval, java.lang.String destinationRealm, long eventTimestamp, int originStateID, Accounting act);	Stops an accounting offline charging session (deprecated)

Table 7. Web service offline charging methods (continued)

Rf accounting method	**Description**
ACAResults*** interimOfflineAccounting(java.lang.String sessionId, int recordNumber, java.lang.String userName, int acctInterimInterval, java.lang.String destinationRealm, long eventTimestamp, int originStateID, Accounting act);	Updates an accounting offline charging session (deprecated)
ACAResults*** eventOfflineAccounting(java.lang.String sessionId, int recordNumber, java.lang.String userName, int acctInterimInterval, java.lang.String destinationRealm, long eventTimestamp, int originStateID, Accounting act);	Processes a one-time offline charging event (deprecated)
Avp**** rawAccounting(Avp[] avp);	Provides a method for the Web services client to manually construct the Diameter packets
* com.ibm.diameter.charging.util.RfAccountingResults ** com.ibm.diameter.charging.util.RfAccountingInfo *** com.ibm.diameter.rf.util.ACAResults **** com.ibm.diameter.packet.Avp	

Rf High-Level API

The Rf interface is a Web service interface to record data for offline charging. Data for the duration of a session or for a single operational event can be tracked and recorded.

The interface is comprised of a number of Web service methods and classes. These classes are used to encapsulate the information in the request arguments and the result values.

Class RfAccountingInfo

The RfAccountingInfo class includes all of the basic informational elements needed to manage an Accounting Session or Event for offline charging.

Usage

The RfAccountingInfo class is a simple representation of the Diameter Accounting Request (ACR) command and includes all of the basic informational elements needed to manage an Accounting Session or Event. The Web service client creates an instance of RfAccountingInfo and must set all required fields before initiating a Web service request.

Depicted in the example and described in the Get and Set Methods tables:

- The sessionId must be set to a globally unique identifier for the accounting session.
- For a single session, the record number for the start request is zero. The record numbers for the interims are 1 through n (where there are n interim messages). The record number for the stop is n+1 (the final message of the session).

- The destinationRealm should align with a realm defined in the routing table of Diameter_Rf.properties. This determines the Charging Collection Function (CCF) to which the Diameter packet will be sent.
- The userName should be included by the Web service client, if known, and uniquely identifies the user in which accounting will be performed at the CCF.
- All other member values are consumed by the CCF and are used to manage the accounting session or to generate Charging Data Records (CDRs). Optional parameters should only be set as needed to ensure the correct generation of CDRs.

Example

```
RfAccountingInfo acctInfo = new RfAccountingInfo();
acctInfo.setSessionId("example.example.com:44321;23433;821;0AB3F12");
acctInfo.setAccountingRecordNumber(0);
acctInfo.setDestinationRealm("example.com");
acctInfo.setUserName("alice@example.com");
RfAccountingResult results = service.startRfAccountingInfo(acctInfo);
```

Get methods

Method	Type	Returns	Description
getSessionId	String	sessionId	The sessionId that is unique for this session. Example: `String sessionId = acctInfo.getSessionId();`
getAccountingRecordNumber	int	accountingRecordNumber	Identifies an accounting message within a session. For a typical single session, the record number for the start is zero, the record number for the interims are 1 through n (where there are n interim messages), and the record number for the stop is n+1 (the final message of the session).
getUserName	String	userName	The private user identity if available in the node. Example: `String username = acctInfo.getUserName();`
getAcctInterimInterval	int	acctInterimInterval	The amount of time in seconds whereby the server wishes to receive updates from the application client. It is the responsibility of the client to provide an interim message at the end of each interval. The client may set this field as a hint to the server.
getDestinationRealm	String	destinationRealm	The administration domain that recognizes the CCF. Example: `String destinationRealm = acctInfo.getDestinationRea`
getEventTimestamp	long	eventTimestamp	The event value for time and MAY be included in ACR and ACA messages to record the time that the reported event occurred. This is specified as milliseconds since January 1, 1970 00:00 UTC. Example: `long eventTimestamp = acctInfo.getEventTimestamp();`

Method	Type	Returns	Description
getOriginStateId	int	originStateId	Holds the value of our Origin-State-Id. Example: `int originStateId = acctInfo.getOriginStateId();`
getServiceInformation	serviceInformation	serviceInformation	Service-specific 3GPP accounting information.

Set methods

Method	Parameter Name	Type	Description
setSessionId	**sessionId**	String	Sets the Session-Id for this message. The Session-Id is unique for a given session. The same Session-Id value should be used in every message of that session. Example: `rfAccountingInfo.setSessionId("sipclient.example.com:`
setAccountingRecordNumber	**accountingRecordNumber**	int	Identifies an accounting message within a session. For a typical single session, the record number for the start is zero, the record number for the interims are 1 through n (where there are n interim messages), and the record number for the stop is n+1 (the final message of the session).
setUserName	**userName**	String	The private user identity if available in the node.
setAcctInterimInterval	**acctInterimInterval**	int	The amount of time in seconds whereby the server wishes to receive updates from the application client. It is the responsibility of the client to provide an interim message at the end of each interval. The client may set this field as a hint to the server.
setDestinationRealm	**destinationRealm**	String	The administration domain that recognizes the CCF.
setEventTimestamp	**eventTimestamp**	long	The event value for time and MAY be included in ACR and ACA messages to record the time that the reported event occurred, in milliseconds since January 1, 1970 00:00 UTC. Example: `rfAccountingInfo.setEventTimestamp(System.currentTime`

Method	Parameter Name	Type	Description
setOriginStateId	**originStateId**	int	Sets the value of our Origin-State-Id. Note that by setting this, your application will indicate to the server that the Origin State has changed (meaning that he will void any existing sessions). And, once you have set it, you must use the same value for all messages that you initiate. Therefore, if you use this method, you must have complete knowledge of the applications running, and they must be well coordinated. Example: `rfAccountingInfo.setOriginStateId(356777);`
setServiceInformation	**serviceInformation**	serviceInformation	Service-specific 3GPP accounting information.

Class RfAccountingResults

The RfAccountingResults class is a simple representation of the Diameter Accounting Answer (ACA) command. It contains all the pertinent information for managing an Accounting Session.

Usage

The Rf Web Service Session and Event methods return an RfAccountingResults object. The Web service client can query the results to further manage an Accounting session. These values are listed in the methods tables in this topic. They are also listed in RFC 3588.

Example

```
RfAccountingResults acaResults = service.startRfAccountingInfo(acctInfo);
acaResults.getAcctInterimInterval();
```

Get methods

Method	Type	Return	Description
getResultCode	int	resultCode	Successful or unsuccessful result codes in the form of an exception. Possible values include: 1xxx (Informational) 2xxx (Success)
getEventTimestamp	long	eventTimestamp	The event time stamp. The event time stamp is returned in time. It may be included in ACR and ACA messages to record the time that the reported event occurred, in milliseconds since January 1, 1970 00:00 UTC. Example: `long eventTimestamp = rfAccountingResults.getEventTi`
getOriginStateID	int	originStateID	The origin state identifier. This Attribute Value Pair (AVP) is optional.
getAcctInterimInterval	int	acctInterimInterval	The interval (in seconds) in which the accounting client should send interim requests after the start of the session.

Set methods

Because the RfAccountingResults object is set by the Diameter Enabler based on results it has received from the CCF, the set methods are rarely used.

Method	Parameter Name	Type	Description
setResultCode	**resultCode**	int	The result code value.
setEventTimestamp	**eventTimestamp**	long	The event time stamp.
setOriginStateID	**originStateID**	int	The origin state identifier.
setAcctInterimInterval	**acctInterimInterval**	int	The interval (in seconds) in which the accounting client should receive interim requests after the start of the session.

Class ServiceInformation (Rf)

This class is a member of RfAccountingInfo and acts as a container for service-specific 3GPP accounting information.

Usage

The 3GPP has separated its vendor-specific AVPs into multiple service-specific charging groups. Contained within ServiceInformation are nested classes that act as holders for service-specific accounting data.

All of the accounting data created by an IMS node is generally located in IMSInformation. However, in some instances, accounting data will need to be generated for multiple services. The Web service client developer simply constructs and sets the service-specific information objects needed to generate accounting information, leaving the irrelevant service objects null.

Within each service-specific class are additional members that represent accounting data that can be processed by a Charging Collection Function (CCF) to produce Charging Data Records (CDRs). The type of data recorded will differ across implementations to meet the needs of the accounting application. All figures and statistics must be in a format that complies with the capabilities of the CCF. In the following example, accounting data is created for the SUBSCRIBE event type.

For detailed information regarding the nesting of IMS accounting data, refer to the Javadoc and the 3GPP specifications.

Example

```
RfAccountingInfo acctInfo = new RfAccountingInfo();
acctInfo.setSessionId("example.example.com:44321;23433;821;0AB3F12");
acctInfo.setAccountingRecordNumber(0);
acctInfo.setDestinationRealm("example.com");
acctInfo.setUserName("alice@example.com");
ServiceInformation svcInfo = new ServiceInformation();
IMSInformation imsInfo = new IMSInformation();
EventType eventType = new EventType();
eventType.setSipMethod("SUBSCRIBE");
imsInfo.setNodeFunctionality(NodeFunctionality.AS);
imsInfo.setEventType(eventType);

svcInfo.setImsInformation(imsInfo);
acctInfo.setServiceInformation(svcInfo);
service.startRfAccountingInfo(acctInfo);
```

Get methods

Method	Type	Return	Description
getImsInformation	IMSInformation	imsInformation	Accounting information generated by an IP Multimedia Subsystem service.
getPsInformation	PSInformation	psInformation	Accounting information generated by a Packet Switched service.
getLcsInformation	LCSInformation	lcsInformation	Accounting information generated by a Location Services service.
getMmsInformation	MMSInformation	mmsInformation	Accounting information generated by a Multimedia Messaging service.
getWlanInformation	WLANInformation	wlanInformation	Accounting information generated by a WLAN service.
getPocInformation	PoCInformation	pocInformation	Accounting information generated by a Push-To-Talk Over Cellular service.
getMbmsInformation	MBMSInformation	mbmsInformation	Accounting information generated by a Multimedia Broadcast and Multicast service.

Set methods

Method	Parameter	Type	Description
setImsInformation	**imsInformation**	IMSInformation	Accounting information generated by an IP Multimedia Subsystem service.
setPsInformation	**psInformation**	PSInformation	Accounting information generated by a Packet Switched service.
setLcsInformation	**lcsInformation**	LCSInformation	Accounting information generated by a Location Services service.
setMmsInformation	**mmsInformation**	MMSInformation	Accounting information generated by a Multimedia Messaging service.
setWlanInformation	**wlanInformation**	WLANInformation	Accounting information generated by a WLAN service.
setPocInformation	**pocInformation**	PoCInformation	Accounting information generated by a Push-To-Talk Over Cellular service.
setMbmsInformation	**mbmsInformation**	MBMSInformation	Accounting information generated by a Multimedia Broadcast and Multicast service.

Deprecated Rf API

This section contains deprecated Rf classes that were used by the deprecated Web service methods.

Class Accounting (Deprecated):

The Accounting class includes methods that get and set information for offline charging.

Usage

The methods can get or set details about the specific transaction. This information is used to determine how much to charge.

This class passes charging information as a single parameter to the Charging Collection Function.

Example

```
Accounting acct = new Accounting(); //create Accounting object
acct.setUserSessionId("a84b4c76e66710@test.acme.com");
acct.setServiceId("12345");
ACAResults results = service.startRfAccounting("MySessionId", "MyUserName", 0,
   "example.com", 2085978496000, 0,acct);
```

Get methods

Method	Type	Returns	Description
getAAppSrvInfo	AppServInfo	aAppSrvInfo	Retrieves information about the accounting servers and identifies information needed to keep track of transactions between networks for charging purposes.
getACauseCode	CauseCode	aCauseCode	Retrieves the cause codes that may have occurred during an accounting request.
getASDPmedia	SDPmedia[ ]	aSDPmedia	Retrieves information about the type of media involved in the accounting transaction.
getASipInfo	SipInfo	aSipInfo	Retrieves SIP information used in accounting to keep track of the parties involved and the type of transaction.
getATrunkGroup	TrunkGroup	aTrunkGroup	Retrieves information which identifies the Public Telephone Switched Network (PSTN).
getAuthorisedQos	String	authorisedQos	Retrieves the quality of service.
getAUUSdata	UUSdata	aUUSdata	Retrieves the information for the User to User Protocol used by accounting to keep track of the amount and type of data involved in the transaction.
getBearerService	String	bearerService	Retrieves the used bearer service for the PSTN.
getGgsnAddress	String	ggsnAddress	Retrieves the IP address of the Gateway GPRS Support Node (GGSN) in the session.
getImsChargingIndentifier	String	imsChargingIndentifier	Retrieves the IMS Charging identifier (ICID) as generated by a node for a SIP session. Example: `String icid = accountingResults.getImsChargingIdentifier();`
getRoleofNode	int	roleofNode	Retrieves the role of the IMS Application Server or the CSCF. Roles include: 0 = ORIGINATING_ROLE 1 = TERMINATING_ROLE 2 = PROXY_ROLE 3 = B2BUA_ROLE
getSdpSessionDescription	String	sdpSessionDescription	Retrieves the session of the SDP data when it is exchanged between the user agents in the SIP transaction.

Method	Type	Returns	Description
getServedPartyIPaddress	String	servedPartyIPaddress	Retrieves the address of the calling party or the called party. **Note:** Only the following attribute lines for charging are recorded: c = connection information Example: `IN IP4 224.2.17.12/127` a = session attribute lines Example: `a = recvonly`
getServiceId	String	serviceId	Retrieves the service the media resource function controller (MRFC) is hosting and is the value of the conference ID.
getUserSessionId	String	userSessionId	Retrieves the SIP session and contains the SIP call ID.

Set methods

Method	Parameters	Type	Description
setAAppSrvInfo	**appSrvInfo**	AppServInfo	Defines information about the accounting servers and identifies information needed to keep track of transactions between networks for charging purposes.
setACauseCode	**causeCode**	CauseCode	Defines the cause codes that may have occurred during an accounting request.
setASDPmedia	**pmedia**	SDPmedia[ ]	Defines information about the type of media involved in the accounting transaction.
setASipInfo	**sipInfo**	SipInfo	Defines SIP information used in accounting to keep track of the parties involved and the type of transaction.
setATrunkGroup	**trunkGroup**	TrunkGroup	Defines information which identifies the PSTN.
setAuthorisedQos	**authorisedQos**	String	Defines the quality of service.
setAUUSdata	**sdata**	UUSdata	Defines the information for the User to User Protocol used by accounting to keep track of the amount and type of data involved in the transaction.
setBearerService	**bearerService**	String	Defines the used bearer service for the PSTN.
setGgsnAddress	**ggsnAddress**	String	Defines the IP address of the GGSN in the session.
setImsChargingIndentifier	**imsChargingIndentifier**	String	Defines the ICID as generated by a node for a SIP session.
setRoleofNode	**roleofNode**	int	Defines the role of the IMS Application Server or the CSCF. Roles include: 0 = ORIGINATING_ROLE 1 = TERMINATING_ROLE 2 = PROXY_ROLE 3 = B2BUA_ROLE

Method	Parameters	Type	Description
setSdpSessionDescription	**sdpSessionDescription**	String	Defines the session of the SDP data when it is exchanged between the user agents in the SIP transaction.
setServedPartyIPaddress	**servedPartyIPaddress**	String	Defines the address of the calling party or the called party. **Note:** Only the following attribute lines for charging are recorded: c = connection information Example: `IN IP4 224.2.17.12/127` a = session attribute lines Example: `a = recvonly`
setServiceId	**serviceId**	String	Defines the service the media resource function controller (MRFC) is hosting and is the value of the conference ID.
setUserSessionId	**userSessionId**	String	Defines the SIP session and contains the SIP call ID.

Class ACAResults (Deprecated):

The ACAResults class is used to transfer information from the Charging Collection Function back to the IMS Application Server application that sent a request using the Rf accounting Web service.

Usage

ACAResults transfers information from the Charging Collection Function using the Rf accounting Web service. When an IMS Application Server application requests a one time charging event, such as eventOfflineAccounting, the Rf accounting Web service will return an ACAResults object. This returns information such as whether the operation completed successfully and other values from the ACA (Accounting Answer). In case of failure, a Java exception will be thrown to the calling application.

Example

```
accountingData = new Accounting(); //create and initialize this object with accounting data
   to pass it on using the eventOfflineAccounting method.
ACAResults myResults = service.eventOfflineAccounting( "MySessionId", "MyUserName", 0,
   "example.com", 2085978496000, 0, accountingData);
```

Get methods

Methods	Type	Returns	Description
getAAct	Accounting	aAct	Returns the Accounting object.
getAcctInterimInterval	int	acctInterimInterval	Retrieves the start time for the session. The acctInterimInterval is returned in seconds.

Methods	Type	Returns	Description
getEventTimestamp	long	eventTimestamp	Retrieves the event time stamp. The event time stamp is returned in time. It may be included in ACR or ACA messages to record the time in milliseconds since January 1, 1970 00:00 UTC. Example: `long eventTimestamp = acaResults.getEventTimestamp();`
getOriginStateID	int	originStateID	Retrieves the origin state identifier. This Attribute Value Pair (AVP) is optional.
getResultCode	int	resultCode	Retrieves the result code value. Possible values include: • 2001: DIAMETER_SUCCESS • 2002: DIAMETER_LIMITED_SUCCESS

Set methods

Methods	Parameter Name	Type	Description
setAAct	**act**	Accounting	Defines the account
setAcctInterimInterval	**acctInterimInterval**	int	Defines the interval at which each update or interim message must be sent.
setEventTimestamp	**eventTimestamp**	long	Defines the event time stamp
setOriginStateID	**originStateID**	int	Defines the origin state identifier
setResultCode	**resultCode**	int	Defines the result code value

Class AppServInfo (Deprecated):

This class provides methods to identify information about the Charging Collection Function. The information can be used to monitor transactions between the networks for charging purposes.

Usage

AppServInfo is used as an object to transfer information about a specific transaction to the Charging Collection Function. This object could contain information about the parties involved in a particular transaction, such as a conference call using cellular devices.

Example

```
Accounting acct = new Accounting();
AppServInfo asi = new AppServInfo();
asi.setApplicationServer("sip:appserver@operator.com"); asi.setAppCalledPartyAddr("sip:joe@operator.com");
acct.setAAppSrvInfo(asi);
```

Get methods

Methods	Type	Returns	Description
getApplicationServer	String	applicationServer	Retrieves the SIP-URLs of the IMS Application Server or servers addressed during the session. For example, for a cell phone communicating with the IMS Application Server, the server's address would be: `sip:appserver@operator.com`.
getAppCalledPartyAddr	String	appCalledPartyAddr	Retrieves the address of another device. Example: `first.last@operator.com`
getMandatoryCapability[ ]	int	mandatoryCapability	Retrieves the mandatory capabilities of the Server Call Session Control Function (S-CSCF), as used by event charging. The operator assigns the unique identifiers for mandatory capabilities.
getOptionalCapability[ ]	int	optionalCapability	Retrieves the optional capabilities of the S-CSCF. This capability is used by event charging.
getServerName[ ]	String	serverName	Retrieves the SIP-URLs used to identify a SIP server. Example: `sip:servername@operator.com`
getOriginatingIOI	String	originatingIOI	Retrieves the inter-operator identifier for the originating network. This is generated by the S-CSCF in the home network of the originating end user.
getTerminatingIOI	String	terminatingIOI	Retrieves the inter-operator identifier for the originating network. This is generated by the S-CSCF in the home network of the terminating end user.

Set methods

Methods	Parameters	Type	Description
setApplicationServer	**applicationServer**	String	Defines the SIP-URLs of the IMS Application Server or servers addressed during the session.
setAppCalledPartyAddr	**appCalledPartyAddr**	String	Defines the address of another device.
setMandatoryCapability[ ]	**mandatoryCapability**	int	Defines the mandatory capabilities of the Server Call Session Control Function (S-CSCF), as used by event charging. The operator assigns the unique identifiers for mandatory capabilities.
setOptionalCapability[ ]	**optionalCapability**	int	Defines the optional capabilities of the S-CSCF. This capability is used by event charging.

Methods	Parameters	Type	Description
setServerName[ ]	**serverName**	String	Defines the SIP-URLs used to identify a SIP server.
setOriginatingIOI	**originatingIOI**	String	Defines the inter-operator identifier for the originating network. This is generated by the S-CSCF in the home network of the originating end user.
setTerminatingIOI	**terminatingIOI**	String	Defines the inter-operator identifier for the originating network. This is generated by the S-CSCF in the home network of the terminating end user.

Class CauseCode (Deprecated):

This class contains methods to get and set cause codes that may have occurred during the accounting process.

Usage

This class passes errors and status codes between transactions.

Example

```
Accounting acct = new Accounting();
CauseCode cc = new CauseCode();
cc.setCausecode(0);
cc.setNodeFunctionality(1);
acct.setACauseCode(cc);
```

Get methods

Methods	Type	Returns	Description
getCausecode	int	Returns one of the following cause codes: 0 = Normal end of session -1 = Successful transaction -2 = SUBSCRIBE dialog -3xx = 3xx Redirection 1 = Unspecified error 2 = Unsuccessful setup 3 = Internal error 4xx = Request failure 5xx = Server failure 6xx = Global failure	Retrieves cause codes
getNodeFunctionality	int	nodeFunctionality	Retrieves the identifier of the node where the causecode was generated

Set methods

Methods	Parameters	Type	Description
setCausecode	**causecode**	int	Defines the cause code

Methods	Parameters	Type	Description
setNodeFunctionality	**nodeFunctionality**	int	Defines the identifier of the node

Class SDPmedia (Deprecated):

This class provides methods for getting or setting information about the media being passed between the device to the server. For example in a music download purchase, this class will identify the type of data in the transaction.

Usage

This class is used to get and set the media information. The Accounting class transfers the media information to the Charging Collection Function.

Example

```
Accounting acct = new Accounting();
SDPmedia[] mediaArray = new SDPmedia[2];
SDPmedia media1 = new SDPmedia();
media1.setGprsChargingId("charging id 1");
media1.setSdpMediaDescription("c=IN IP4 134.134.157.81");
media1.setSdpMediaName("m=video51372");
mediaArray[0] = media1;
SDPmedia media2 = new SDPmedia();
media2.setGprsChargingId("charging id 2");
media2.setSdpMediaDescription("c=IN IP4 134.134.157.34");
media2.setSdpMediaName("m=video51372");
mediaArray[1] = media2;
acct.setASDPmedia(mediaArray);
```

Get methods

Method	Type	Returns	Description
getSdpMediaName	String	sdpMediaName	Retrieves the media name. The name starts with `m =`, such as: `m = `*`media port transport fmt list`* Where: *media* represents the type of media, such as audio, video, application, data, and control *port* represents the transport port that receives the media stream *transport* represents the protocol used, such as UDP *fmt list* represents the list of media formats
getSdpMediaDescription	String	sdpMediaDescription	Retrieves the connection information for media transactions.
getGprsChargingId	String	gprsChargingId	Retrieves the IMS charging identifier.

Set methods

Method	Parameter	Type	Description
setSdpMediaName	**sdpMediaName**	String	Sets the name of media.

Method	Parameter	Type	Description
setSdpMediaDescription	**sdpMediaDescription**	String	Sets the connection information for media transactions.
setGprsChargingId	**gprsChargingId**	String	Sets the IMS charging identifier.

Class SipInfo (Deprecated):

This class includes methods that get and set information for the SIP information protocol that is used by the Accounting class.

Usage

The methods can get or set details about the specific transaction, such as what type of data and how much data was transferred. This information is used to determine how much to charge.

This class transfers SIP requests. The Accounting class passes the information to the Charging Collection Function.

Example

```
Accounting acct = new Accounting();
SipInfo si = new SipInfo();
si.setCalledPartyAddress("sip:alice@example.com");
si.setCallingPartyAddress("sip:bob@example.com");
si.setContentDisposition("session;handling=optional");
si.setContentLength("142");
si.setContentType("application/sdp");
si.setEvent("subscribe header");
si.setSipMethod("INVITE");
si.setSipRequest("1141094426");
si.setSipResponse("114109943");
acct.setASipInfo(si);
```

Get methods

Method	Type	Returns	Description
getCalledPartyAddress	String	calledPartyAddress	Retrieves the address for the established session. Example: `sip@amy.example.com`
getCallingPartyAddress	String	callingPartyAddress	Retrieves the address of the third party initiating the session. Example: `Public User ID: sip@bob.example.com`
getContentDisposition	String	contentDisposition	Indicates how the message body is interpreted. Example: `contentDisposition="render"`
getContentLength	String	contentLength	Retrieves the size of the message body. There are no size limitations. Example: `12345678`

Method	**Type**	**Returns**	**Description**
getContentType	String	contentType	Retrieves the media type. Possible values are application, html, sdp, or text. Example: `contentType="application"`
getEvent	String	event	Retrieves content of the event handler used in SUBSCRIBE and NOTIFY functions.
getSipMethod	String	sipMethod	Retrieves the name of SIP method. Example: `INVITE`
getSipRequest	String	sipRequest	Retrieves the time in UTC format of the initial SIP request. Example: `Invite`
getSipResponse	String	sipResponse	Retrieves the time in UTC format of the response to the initial SIP request. Example: `200 OK`

Set methods

Method	**Parameters**	**Type**	**Description**
setCalledPartyAddress	**calledPartyAddress**	String	Defines the address for the established session.
setCallingPartyAddress	**callingPartyAddress**	String	Defines the address of the third party initiating the session.
setContentDisposition	**contentDisposition**	String	Defines how the message body is interpreted.
setContentLength	**contentLength**	String	Defines the size of the message body. There are no size limitations.
setContentType	**contentType**	String	Defines the media type. Possible values are application, html, sdp, or text.
setEvent	**event**	String	Defines content of the event handler used in SUBSCRIBE and NOTIFY functions.
setSipMethod	**sipMethod**	String	Defines the name of SIP method.
setSipRequest	**sipRequest**	String	Defines the time in UTC format of the initial SIP request.
setSipResponse	**sipResponse**	String	Defines the time in UTC format of the response to the initial SIP request.

Class TrunkGroup (Deprecated):

This class provides methods to get and set the identify the Public Switched Telephone Network (PSTN) that are involved in a transaction.

Usage

This class organizes information about the PSTN. The Accounting class passes the information to the Charging Collection Function.

Example

```
Accounting acct = new Accounting();
TrunkGroup tg = new TrunkGroup();
tg.setIncomingTrunkGroupId("incoming trunkid");
tg.setOutgoingTrunkGroupId("outgoing trunkid");
acct.setATrunkGroup(tg);
```

Get methods

Method	Type	Returns	Description
getIncomingTrunkGroupId	String	incomingTrunkGroupId	Retrieves the incoming PSTN leg
getOutgoingTrunkGroupId	String	outgoingTrunkGroupId	Retrieves the outgoing PSTN leg

Set methods

Variable name	Parameter	Type	Description
setIncomingTrunkGroupId	**incomingTrunkGroupId**	String	Defines the incoming PSTN leg
setOutgoingTrunkGroupId	**outgoingTrunkGroupId**	String	Defines the outgoing PSTN leg

Class UUSdata (Deprecated):

This class includes methods that get and set information for the User-to-User protocol that is used by Accounting.

Usage

The methods can get or set details about the specific transaction, such as what type of data and how much data was transferred. This information can be used to determine how much to charge.

This class organizes information about User-to-User protocol data. The Accounting class passes the information to the Charging Collection Function.

Example

```
Accounting acct = new Accounting();
UUSdata data = new UUSdata();
data.setAmountOfUUSdata("435");
data.setDirection(0);
data.setMimeType("image/jpeg");
acct.setAUUSdata(data);
```

Get methods

Method	Type	Returns	Description
getAmountOfUUSdata	String	amountOfUUSdata	Retrieves the amount of User-to-User data in the body of the SIP message.
getMimeType	String	mimeType	Retrieves information about the sent User-to-User data such as the AVP/MIME type

Method	Type	Returns	Description

getDirection	int	direction	Retrieves direction that User-to-User data travels
			UPLINK=0
			DOWNLINK=1

Set methods

Method	Parameters	Type	Description
setAmountOfUUSdata	**amountOfUUSdata**	String	Defines the amount of User-to-User data in the body of the SIP message
setMimeType	**mimeType**	String	Defines AVP/MIME type for the User-to-User data

setDirection	**direction**	int	Defines the direction that User-to-User data travels
			UPLINK=0
			DOWNLINK=1

Rf Raw API

The Rf Raw API is a more flexible method to send custom session and event requests that require more knowledge of the underlying protocol.

The high-level Rf Web service methods are intended to simplify the programming requirements to create and issue accounting requests. They follow the Augmented Mackus-Naur Form (ABNF) structures defined by the 3GPP and Internet Engineering Task Force (IETF). However, there are times when this ABNF is not valid, including:

- When new versions of the 3GPP or IETF specifications are available and the new ABNF does not match the old one.
- When an application needs to supplement the ABNF using Attribute Value Pairs (AVPs) that are not defined by the 3GPP or IETF.

When these changes are required, then the high-level Rf Web service methods will not be usable to meet the new ABNF needs. Instead, the rawAccounting() Web service method must be used.

Usage

The rawAccounting() Web service allows the developer using this method to completely control the order, type, and content of each AVP to be included in the Diameter Accounting Request (ACR). These AVPs can be defined by the IETF, the 3GPP, or by any other vendor. The difference is that the application server application must construct an array of these AVPs before invoking the rawAccounting() method.

When the rawAccounting() Web service returns, the value returned is an array of the AVPs received from the Charging Collection Function in the Accounting Answer (ACA) packet. It is then up to the Application Server Application to parse the AVP array and retrieve any of the information that it requires.

The rawAccounting() Web service is largely a pass-through method. It encapsulates the array of AVPs in the request and places them in a Diameter ACR frame. When the ACA is received, it extracts the AVPs from the ACA, places them into an array, and returns them to the caller. There is very little checking performed by the rawAccounting() Web service.

AVPs

The IBM WebSphere Diameter Enabler Component communicates with a Charging Collection Function (CCF) through the Diameter Protocol. Each Diameter Packet contains a list of AVPs, which are used to transfer information between Diameter peers. The Attribute determines the type of information while the Value determines its contents. The IETF has defined a variety of AVPs to be used to manage an accounting session and the 3GPP has defined additional AVPs to report accounting information to a CCF.

Note: The format of an AVP header is defined in RFC 3588.

Contained within the AVP header, the AVP Code in combination with the Vendor-ID uniquely identifies the AVP. AVPs defined in the base specification (RFC 3588) use a Vendor-ID of 0 and AVPs defined by the 3GPP use a Vendor-ID of 10415. The AVP flags (VMPrrrrr) further classify the AVP. If the V(endor-Specific) bit is set, the AVP is defined by a vendor other than the IETF. If the M(andatory) bit is set, the AVP MUST be understood by the Diameter peer. The P bit is an obsolete security bit, and should always be zero.

Using Helper Classes

There are several classes provided with the Toolkit to assist you in developing applications that use the Raw interface. These classes are packaged into two main JAR files:

- DHADiameterPacket.jar
- DHADiameterChargingUtil.jar

DHADiameterPacket.jar contains the classes that make up an AVP. You use the com.ibm.diameter.packet.Avp class when the AVP you wish to send is a base AVP (defined by the IETF). You use the com.ibm.diameter.packet.VsAvp class when the AVP that you wish to send is a vendor-specific AVP. (AVPs defined by the 3GPP are vendor-specific AVPs.)

Note: The internal data or value that the AVP contains is strongly typed. The typing is defined for each AVP. To handle different internal data types, the Diameter Helper classes use an abstract type of com.ibm.diameter.packet.AvpValueUtil. The actual types extend AvpValueUtil and are data types that match those defined by the Diameter specifications. They include:

- AvpValueUtilGrouped
- AvpValueUtilOctetString
- AvpValueUtilUnknown
- AvpValueUtilUnsigned32
- AvpValueUtilUnsigned64
- AvpValueUtilUTF8String

These are fairly self explanatory based on their names. For example, the AvpValueUtilUnsigned32 contains a 32 bit integer. However, the actual value in java is signed, so you must exercise care in using the value especially if you are performing arithmetic with it. You would need to take the hex value and convert it to a long through a logic operations such as an ’AND’.

There are a two AVP types that need some elaboration. The first is the Grouped AVP. Any time the AVP you are using is of type Grouped, it contains other AVPs. The AvpValueUtilGrouped helper class contains a vector of AVPs.

The other AVP type is AvpValueUtilUnknown. This AVP type is used by the Diameter Enabler when it is attempting to parse the data stream, but does not know the definition for a given AVP. A case when this might happen is when a server defines its own, custom Vendor Specific AVP outside of the IETF or 3GPP. The Diameter Enabler, when it receives an AVP that it does not recognize will store all of the data for that AVP as a byte array or octet string. So, if the custom AVP was of type unsigned 32, then the Diameter Enabler would read it into an array with a length of 4 bytes and place this value in an AvpValueUtilUnknown object in the AVP.

Thus, when you are using the Raw interface to create a custom frame and expect to receive back custom AVPs. The AVPs unknown to the Diameter Enabler will have AvpValueUtilUnknown values in them. The application must then convert that byte array to the appropriate type.

Any AVP that is unknown to the Diameter Enabler will receive this handling including custom Grouped AVPs. Even if some of the AVPs that a Grouped AVP holds are known to the Diameter Enabler, they will be included as data in the byte array that contains the payload of the owning Grouped AVP.

For more information on the helper classes and their methods, refer to the Javadoc. These classes are provided as part of the IMS Connector CD. They are also included in the IMS Tooling packages associated with the IMS Connector.

Defining Your Own AVPs

You can define any AVP that you wish to send in the ACR message. Most of the AVPs defined in the IETF RFCs 3588 and 4006 are already defined within the Diameter Enabler helper classes. You can simply create an AVP by providing the data to an AvpFactory method. For example:

```
Avp originRealmAvp = AvpFactory.createOriginRealmAvp("asclient.example.com");
```

This example creates an Origin-Realm AVP whose value is asclient.example.com. Many of these AVPs are vendor-specific for IMS with a vendor ID of 10415. Here is an example of the way in which ChargingAvpFactory is used:

```
Avp exponentAvp = ChargingAvpFactory.createExponentAvp(-2);
```

The AVPs that you are able to create using the AvpFactory and ChargingAvpFactory are limited to those defined by the Diameter Enabler. You may want to create your own AVPs that are not included in the Diameter Enabler.

You can do this using the helper classes: com.ibm.diameter.packet.Avp and com.ibm.diameter.packet.AvpValueUtil<type>. For example:

```
AvpValueUtil numberOfMembersValueUtil = new AvpValueUtilUnsigned32(20);
Avp numberOfMembersAvp = new VsAvp(NUMBER_OF_MEMBERS,
     Avp.VON_MON_POFF,numberOfMembersValueUtil, VENDOR_ID_COMPANYX);
```

This example creates a new vendor-specific AVP of type Unsigned32 and loads it with a value of 20. It also creates the AVP with the V-bit (vendor -specific) set to 1, the M-bit (Mandatory) set to 1, and the P-bit set to 0. Finally, the Vendor ID is set to COMPANYX's vendor ID. The vendor ID should be listed in the enterprise-numbers2 on the Internet Assigned Numbers Authority site.

Receiving and Processing the Results of Your Raw Request

You create AVPs when you wish to send a data element to the CCF. You receive back the AVPs that the CCF sends in its ACA. The result is returned in an Array of AVPs that occur in the order in which the AVPs were received with array[0] being the first AVP received. The type of data that is held within the AVP is either that defined by the IETF or 3GPP if the AVP is formally defined in the Rf definition. If the AVP is not in any of these definitions, then the data type will be AvpValueUtilUnknown.

Whenever you receive an AVP with data of type AvpValueUtilUnknown, the Diameter Enabler does not have a definition of the AVP in its dictionary. Your application must convert the byte array into data of a type that is usable by your application. Simple types, such as Unsigned32, are straight forward. However, the Grouped type means that AVPs including their AVP headers will be included in the byte array. And because there can be any number of nesting levels, Grouped AVPs will need some additional parsing algorithm to convert them from an array of bytes into an array of AVPs or other data elements.

In general, if you wish to retrieve information from the reply frame, you will walk through the list of AVPs received, pull the one of interest, get its AvpValueUtil, and invoke the getAvpValue() method of the AvpValueUtil.

Limitations of the Raw Interface

If you create an AVP that is already defined by the Diameter Enabler, your AVP will be transmitted with the type, values, and AVP header bits that you have defined. However, if the same AVP is received by the Diameter Enabler in the reply, the Diameter Enabler's definition will be applied when creating the AVP.

Also, an incorrectly created AVP may affect the data stream and cause an Out-Of-Sync error to occur. If when transmitting, the number of bytes expected does not match the length parameters, the packet will be malformed. An error such as this may cause the other side of the connection to determine that it no longer knows where it is in the data stream. This generally results in the connection being dropped and re-established. For this reason, you need to secure and limit the access to these data services to trusted applications.

Example

```
DiameterRfService_SEIServiceLocator locator = new DiameterRfService_SEIServiceLocator();
DiameterRfService_SEI service = locator.getDiameterRfService(endpoint);
 //create an Avp Array
      Avp[] avps = new Avp[7];
 //create Session-Id avp
      Avp avp = new Avp(AvpConstants.SESSION_ID, Avp.VOFF_MON_POFF, new AvpValueUtilUTF8String("session"));
      avps[0]=avp;
 //create Origin-Host avp
      avp = new Avp(AvpConstants.ORIGIN_HOST, Avp.VOFF_MON_POFF, new AvpValueUtilUTF8String(originHost));
      avps[1]=avp;
 //create Origin-Realm avp
      avp = new Avp(AvpConstants.ORIGIN_REALM, Avp.VOFF_MON_POFF, new AvpValueUtilUTF8String(originRealm));
      avps[2]=avp;
 //create Destination-Realm avp
      avp = new Avp(AvpConstants.DESTINATION_REALM, Avp.VOFF_MON_POFF, new AvpValueUtilUTF8String(destinationRealm));
      avps[3]=avp;
 //create Accounting-Record-Type avp
      avp = new Avp(AvpConstants.ACCOUNTING_RECORD_TYPE, Avp.VOFF_MON_POFF, new AvpValueUtilUnsigned32(2)); //START_RECORD
      avps[4]=avp;
 //create Accounting-Record-Number avp
      avp = new Avp(AvpConstants.ACCOUNTING_RECORD_NUMBER, Avp.VOFF_MON_POFF, new AvpValueUtilUnsigned32(0)); //0 because this is a start record
      avps[5]=avp;
 // create User_Name avp
```

```
      avp = new Avp(AvpConstants.USER_NAME, Avp.VOFF_MON_POFF, new AvpValueUtilUTF8String("Neo"));
      avps[6]=avp;
 //Send the Web Service Request
Avp[] result = service.rawAccounting(avps)
```

Ro online charging Web service

Ro online charging Web service provides an IMS Application Server application with a Diameter messaging interface to enable the application to send credit control messages to online charging servers. The IMS Application Server application is referred to as a client of the Web service application.

The IMS Application Server communicates with the online charging server through Ro online charging Web service using either a session or event online charging method as shown here:

- Session charging - Charging for a session that takes place over a period of time.
  1. Initial: Starts an online charging session.
  2. Update: Periodically updates the online charging session.
  3. Terminate:Stops the online charging session.
- Event charging - Charging transaction for a single operation.

The Ro online charging Web service also provides subscription and notification operations for an IMS Application Server to subscribe to receive notifications when the Charging Trigger Function (CTF) (the IMS Application Server application) must reauthenticate with the Online Charging System (OCS) or when a reauthorization subscription expires.

The choice of using session charging versus event charging will depend on the applications using this interface. The actual accounting functionality is based on other network elements, such as billing, and is not part of the Ro online charging Web service. The Ro interface defines several high level Web service methods used for online charging including sendCCInitial(), sendCCUpdate(), and sendCCTermination() for session handling and event charging with unit reservation. The sendCCDirectDebit(), sendCCRefund(), and getCCServicePriceEnquiry() are used for event charging without unit reservation. And, the subscribeCCReAuth(), unsubscribeCCReAuth, and notifyCCReAuth() are used for subscribe and notify requests. The notify request (notifyCCReAuth()) is sent as a Web service request from the Diameter Enabler to the Application.

Application overview

The following is a brief overview of the applications used with Diameter:

- IMS Application Server Application: The IMS Application Server Application uses WSDL and/or helper classes to prepare a Web service request to be sent to one of the Diameter Enabler Web service applications.

  If the Diameter Enabler Web service supports notifications, the IMS Application Server application must use the WSDL and/or helper classes to implement the client Web service that receives notifications. The URL of the client Web service is provided to the Diameter Enabler Web service through the subscribeCCReAuth() request.
- Diameter Enabler base: The Diameter Enabler is capable of receiving Web service requests and converting them into Diameter request packets. The Diameter Enabler receives the Web service request, validates its contents, converts it to a Diameter request packet, and routes it to the correct Diameter Server. The

Diameter Server responds with a Diameter response packet, which is returned as a Web service response object to the IMS Application Server application that performed the request.

  The Diameter Enabler is also capable of receiving notifications requests from a Diameter Server. The Diameter Enabler validates the contents of the Diameter request packet and responds to the peer with a Diameter answer packet. If the IMS Application Server Application has created a subscription to receive the notifications, the Diameter Enabler opens an outbound Web service request to a Web service endpoint provided in the subscription data, and passes the reauthorization notification information accordingly.
- Diameter Server: The Diameter Server is a peer connection to the Diameter Enabler base and is capable of sending and receiving Diameter request and answer packets.

Application deployment

Ro online charging Web service is deployed in a WebSphere Application Server environment. Prior to installing Ro online charging Web service, you must install Diameter Enabler base. Any application with the appropriate access and security may invoke these functions. The supported transport is SOAP over HTTP. The Ro online charging Web service is stateless and supports multiple simultaneous calls.

Ro online charging Web service call flows

Ro online charging Web service call flows describe how the application transmits and receives accounting packets (credit control request (CCR), credit control answer (CCA)) and reauthorization packets (Reauthorization Request (RAR), Reauthorization Answer (RAA)). It also describes how the Diameter Enabler base interacts with the Application client and the Online Charging System (OCS).

Diameter Enabler base supports multiple realms. The internal routing table controls which realm a packet is sent to.

The Ro online charging Web service application validates the parameters received from the Web service interface; then, builds a Diameter packet (CCR) using the RoChargingInfo parameter. The Ro Web service application passes the Diameter packet to the Diameter Enabler base.

Ro online charging Web service uses two key flows (request/reply and subscribe/notify) to handle charging transactions:

- Request/Reply flow:
  1. An IMS Application Server, the SIP application, invokes one of the Ro online charging Web service methods, such as sendCCInitial.
  2. The Ro online charging Web service application validates the parameters received from the Web service interface; then, builds a Diameter packet (CCR) using the RoChargingInfo parameter that is sent to the Diameter Enabler base.
  3. The Diameter Enabler base sends the Diameter request message to the OCS.
  4. The OCS receives the request message, updates the number of ″used units″ from the subscriber’s account, reserves quota for the subscriber, and returns the result (CCA) to the Diameter Enabler base.
  5. The Diameter Enabler base passes the response to Ro online charging Web service application.

6. The Ro online charging Web service application validates this reply and checks for any violations; then, once verified, extracts the information from the CCA and builds a result using the RoChargingResults object that is returned to the SIP application that invoked the Web service request.
7. The SIP application interprets the result and processes the charging information.

- Subscribe/Notify flow:
  1. An IMS Application service invokes the subscribeCCReAuth Web service method, specifying a sessionId and realm, for which it wants to receive reauthorization notifications. In addition, the caller specifies the callbackURL of the application that implements the DiameterRoNotify Web service interface.
  2. The Ro online charging Web service application validates the request and adds a subscription record to the data store, which will remain active for the expiration duration specified in the subscription request.
  3. The OCS sends a RAR to the Diameter Enabler to request that a user is reauthorized for the charging session. The RAR is validated for correctness and a ReAuthInfo object is created. The Diameter Enabler responds to the OCS with a RAA message.
  4. The Ro online charging Web service application searches the subscription database for the Session-Id and Origin-Realm found in the RAR.
  5. If the subscription is active, the callbackURL is retrieved and the Diameter Enabler invokes the notifyCCReAuth Web service request, sending the ReAuthInfo object to the remote DiameterRoNotify Web service.

Ro online charging Web service methods

Ro online charging Web service defines several external Web service methods used for online, session and event charging.

According to RFC 3588 the **sessionID** is an agreed upon format between the IMS Application Server and the server. The same **sessionID** is expected to be used for each request associated with a session. The session is terminated when a stop message is sent using sendCCTermination.

Web service charging methods

The following methods can be used to send online charging information through the Web service interface. Ro online charging Web service supports the following external methods that are available to the IMS Application Server applications. For detailed information on method signatures and descriptions, refer to the com.ibm.diameter.ro.DiameterRoService_SEI and com.ibm.diameter.ro.DiameterRoNotifyService_SEI sections in the Javadoc.

Table 8. Web service online charging methods

Ro charging method	**Description**
RoChargingResults* sendCCInitial(RoChargingInfo** roChargingInfo);	Starts an online charging session
RoChargingResults* sendCCUpdate(RoChargingInfo** roChargingInfo);	Updates an online charging session
RoChargingResults* sendCCTermination(RoChargingInfo** roChargingInfo);	Stops an online charging session

Table 8. Web service online charging methods (continued)

Ro charging method	Description
RoChargingResults* sendCCDirectDebit(RoChargingInfo** roChargingInfo);	Performs an immediate subtraction of credits from the subscriber account
RoChargingResults* sendCCRefund(RoChargingInfo** roChargingInfo)	Performs an immediate addition of credits to the subscriber account
RoChargingResults* getCCServicePriceEnquiry(RoChargingInfo** roChargingInfo);	Retrieves cost information for a specific service
void subscribeCCReAuth(java.lang.String sessionId, java.lang.String destinationRealm, java.net.URI callbackUri, int subscriptionDuration, java.lang.String userid, java.lang.String password);	Subscribes to receive reauthorization requests for a specific sessionId
void unsubscribeCCReAuth(java.lang.String sessionId, java.lang.String destinationRealm, java.net.URI callbackUri, java.lang.String userid, java.lang.String password);	Unsubscribes to receive reauthorization requests for a specific sessionId
void notifyCCReAuth (RoReAuthInfo*** roReAuthInfo);	Informs the Charging Trigger Function (CTF) that there has been a change at the server and that the CTF needs to reauthenticate with the Online Charging System prior to continuing to provide the service
void notifySubscribeExpired (java.lang.String sessionId, java.lang.String destinationRealm);	Notification sent by the Diameter Enabler when a subscription created by the subscribeCCReAuth() method has expired. The subscription duration is specified by an argument in the subscribeCCReAuth() request. To avoid receiving this notification, the application can unsubscribe using the unsubscribeCCReAuth() method when the session completes.
Avp**** sendCCRaw(Avp[] avp);	Sends a raw Diameter message using the Ro interface.
* com.ibm.diameter.charging.util.RoChargingResults ** com.ibm.diameter.charging.util.RoChargingInfo *** com.ibm.diameter.charging.util.RoReAuthInfo **** com.ibm.diameter.packet.Avp	

Ro High-Level API

The Ro interface is a Web service interface to allow participation in online charging transactions. Data for the duration of a session or for a single operational event can be tracked and recorded.

The interface is comprised of a number of methods and helper classes that are necessary to use those methods.

Class RoChargingInfo

The RoChargingInfo class includes all of the basic informational elements needed to manage Session, Event or Immediate Event for online charging.

Usage

The RoChargingInfo class includes all of the basic informational elements needed to manage online charging for Events or Sessions. The online client (Charging Trigger Function (CTF)) requests resource allocation and reports credit control usage information to the Online Charging System (OCS). For online charging, the Diameter Credit Control Application (DCCA), defined in RFC 4006, is used with extensions based on the 3GPP IMS specifications. The following are three cases for control of user credit for online charging:

- Immediate Event Charging (IEC) - the Credit Control Request (CCR) contains a CC-Requested-Type of ″EVENT_REQUEST″. This request is used to make an immediate credit or debit to the subscriber’s account.
- Event Charging with Unit Reservation (ECUR) - a reservation is made for a single event transaction as thought it were a session transaction. The CCR request contains a CC-Request-Type set to INITIAL for the reservation; then, sends a request type set to TERMINATION_REQUEST when the transaction is successfully delivered.
- Session Charging with Unit Reservation (SCUR) - is used for credit control of sessions and uses the CC_Request-Type set to INITIAL_REQUEST, UPDATE_REQUEST, or TERMINATION_REQUEST.

The RoChargingInfo class, a general purpose class, provides a data container for all individual and grouped Attribute Value Pair (AVP) information that is required for a CCR request. In some cases the definition of the AVP includes grouped AVP members that are typically not included in the CCR. This conflict between the AVP definition and the CCR definition is handled in the following way:

- The RoChargingInfo class is constructed based on AVPs that are defined with a consistent AVP structure. They do not hold different definitions for different applications. However, many of the fields are optional, and are not appropriate for the CCR request. The actual format of the content and specific AVPs that are to be included in the CCR will be defined between the Application that sends the CCR and the Server that receives and interprets the CCR request.
- The RoChargingInfo allows for you to send all optional AVPs defined in the Augmented Mackus-Naur Form (ABNF), but it also allows you to send some AVPs that are not recommended for the CCR request. You should only set the fields that the OCS is expecting in each CCR request. The Diameter Enabler verifies that the minimum required information is included in the RoChargingInfo object that is passed through the Web service request. However, it will allow any of the additional fields to be set. It is up to the application developer to verify the contents required to satisfy the OCS beyond those defined in the specifications. Setting fields without an understanding of what is required by the OCS will cause inconsistent results.
- Conversely, when the Diameter Enabler receives a Credit Control Answer (CCA) response, it also allows additional AVPs to be included in the frame. The Diameter Enabler verifies the minimum requirement of the frame and then passes the results back to the application. If the minimum ABNF requirements are not met, the Diameter Enabler returns an Error message to the OCS, will post a message in the logs, and will return an exception back to the application. If the OCS returns a CCA that holds optional non-mandatory AVPs that are recognized, their values will be set in the RoChargingResults. However, if there are AVPs that are not recognized, these AVPs will be discarded by the Diameter Enabler and only the recognized AVP data will be returned to the Application.

Note: If the OCS has defined a custom set of AVPs or custom framing in either what it expects for the CCR or what it returns in the CCA, then the application must be written to use the Raw interface. The Raw interface allows custom AVPs and framing to be used to override the high level APIs provided for the Ro transactions.

Example

```
RoChargingInfo roChargingInfo = new RoChargingInfo();

  //Set the Session-Id to a unique identifier.
  roChargingInfo.setSessionId("FQDNServerName:servicexyz:session00001");
  //Set the Destination-Realm of the OCS, which should match a route name in the Diameter_Ro.properties file.
  roChargingInfo.setDestinationRealm("emulators.example.com");
  //Set the CC-Request-Number to 0 for an initial request.
  roChargingInfo.setCcRequestNumber(0);
  roChargingInfo.setServiceContextId("12345@example.com");

  //Set the subscription id to identify the end user's subscription with the OCS.
  SubscriptionId subId = new SubscriptionId();
  subId.setSubscriptionIdData("id0001");
  subId.setSubscriptionIdType(SubscriptionIdType.END_USER_IMSI);
  roChargingInfo.setSubscriptionId(subId);

  MultipleServicesCreditControl mscc = new MultipleServicesCreditControl();

  //Request quota in time Units
  RequestedServiceUnit rsu = new RequestedServiceUnit();
  rsu.setCcTime(60);
  mscc.setRequestedServiceUnit(rsu);
  roChargingInfo.setMultipleServicesCreditControl(mscc);
```

Set methods

Method Name	Type	Example values/comments
setSessionId()	String	Sets the SessionId that is unique for this session. Example: `roChargingInfo.setSessionId("sipclient.example.com:33041;23432;`
setOriginHost()	String	The fully qualified domain name of the host initiating this request. **Note:** This value is overridden by all Web service requests except sendCCRaw().
setOriginRealm()	String	The fully qualified domain name of the realm where this request originated. **Note:** This value is overridden by all Web service requests except sendCCRaw().
setDestinationRealm()	String	Holds the administration domain that recognizes the OCS. Example: `roChargingInfo.setDestinationRealm("example.com");` **Note:** This value will be used to route the request to the proper OCS.
setAuthApplicationID()	Integer	The value for the Credit Control Application is 4. **Note:** This value will be overridden if using a non-raw method.
setServiceContextId()	String	The specific request document that this request follows: For IMS charging: ″32260@3gpp.org″ Example: `roChargingInfo.setServiceContextId("32260@3gpp.org");`

Method Name	Type	Example values/comments
setCCRequestType()	Integer	This Enumeration contains the following: 1. Initial Request 2. Update Request 3. Termination Request 4. Event Request **Note:** This value will be overridden if using a non-raw method.
setCCRequestNumber()	Integer	Indicates the numbered request within a session. This should be set to 0 for Request Types of Initial and Event. It should be incremented by one for each subsequent request within a session. It is the caller's responsibility to set the proper number.
setDestinationHost()	String	The fully qualified domain name of the Destination Host. This is generally not specified when using realm routing.
setUserName()	String	Name of the subscriber.
setOriginStateId()	Integer	Holds the value to track the incremented value of possible times the client has loss the state or possible reboot has occurred.
setEventTimestamp()	Long	Holds the event value for time and MAY be included in a CCR message to record the time that the reported event occurred, in milliseconds since January 1, 1970 00:00 UTC. This timestamp is converted to the SNTP time format when sending a request. **Note:** This time is based on Java time. Example: `roAccountingInfo.setEventTimestamp(new Long(system.currentTimeMill`
setSubscriptionId()	SubscriptionId	Grouped AVP used to specify the end user, and the format in which the data specifying the end user is presented. For additional information on creating and populating the SubscriptionId, refer to the Javadoc.
setTerminationCause()	Integer	The session was terminated for one of several reasons. 1. LOGOUT 2. SERVICE_NOT_PROVIDED 3. BAD_ANSWER 4. ADMINISTRATIVE 5. LINK_BROKEN 6. AUTH_EXPIRED 7. USER_MOVED 8. SESSION_TIMEOUT
setRequestedAction()	Integer	This value defines the type of action if the CCRequestType is EVENT. 1. Direct Debiting 2. Refund Account 3. Check Balance 4. Price Enquiry

Method Name	Type	Example values/comments
setMultipleServicesIndicator()	Integer	The following values are defined for the Multiple-Services-Indicator AVP: **0: MULTIPLE_SERVICES_NOT_SUPPORTED** The client does not support independent credit-control of multiple services within a session or sub-session **1: MULTIPLE_SERVICES_SUPPORTED** The client supports independent credit-control of multiple services within a session or sub-session
setMultipleServicesCreditControl()	MultipleServicesCreditControl	Contains all fields for quota management. (Refer to Javadoc.)
setUserEquipmentInfo()	UserEquipmentInfo	Contains information about the equipment being used by the subscriber. (Refer to Javadoc.)
setServiceInfo()	ServiceInfo	Holds service-specific 3 GPP accounting and charging information. (Refer to Javadoc.)

Class RoChargingResults

The RoChargingResults class is created from the contents of the Credit Control Answer (CCA) response and is returned back to the application that has invoked one of the following methods: `sendCCInital()`, `sendCCUpdate()`, `sendCCTermination()`, `getCCBalanceCheck()`, `sendCCDirectDebit()`, and `getCCServicePriceEnquiry()`.

Usage

The RoChargingResults class is a representation of the information contained in the AVPs of the CCA message received by the Diameter Enabler in response to the Credit Control Request (CCR) that it sent out. The Diameter Enabler extracts the set of AVPs held in the CCA and populates the fields of the RoChargingResults class with the value information from each AVP. For example, if the CCA contains a Cost-Information AVP, then the field in the RoChargingResults object will have the Cost-Information variable set to the value of the Cost-Information AVP. Because the Cost-Information is a grouped AVP, the data object assigned to the variable is of type ″CostInformation″ which also contains fields representing AVPs that are included inside the Cost-Information AVP.

An application can interrogate the contents of the RoChargingResults class with simple getter methods. If an AVP is in the CCA frame directly, then it is included as an instance variable of the RoChargingResults class. If the AVP is included in a grouped AVP (for example, Cost-Information), then the grouped AVP is represented as an instance variable of the RoChargingResults class while the AVPs that it holds (for example, UnitValue, CurrencyCode, and CostUnit) are represented by an instance variable of the class representing the grouped AVP (the Cost-Information class).

Example

Session:

```
RoChargingResults roChargingResults = svc.sendCCInitial(roChargingInfo);

  MultipleServicesCreditControl mscc = roResult.getMultipleServicesCreditControl();

  //The client can request 30 more seconds of Time Quota before Reauthorization is required.
  Integer timeQuotaThresh = mscc.getTimeQuotaThreshold(); //Example value = 30.

  //The client has been granted 60 seconds of Time Quota.
  GrantedServiceUnit gsu = mscc.getGrantedServiceUnit();
  Integer grantedTimeUnits = gsu.getCcTime();  //Example value = 60
```

```
//Unit cost is 3 US Dollars per minute
CostInformation costInfo = roResult.getCostInformation();
String costUnit = costInfo.getCostUnit();  //Example value = "Access to this service is 3.00 a minute."
Integer currencyCode = costInfo.getCurrencyCode();  //Example Value = 840 (USD)
UnitValue unitValue = costInfo.getUnitValue();
Integer exponent = unitValue.getExponent();  //Example value = 1
```

Get methods

Method Name	Type	Returns	Example values/comments
getSessionId()	String	SessionId	Gets the SessionId for this session
getResultCode()	Integer	getResultCode	1xxx - Information 2xxx - Success Others - handled through exceptions.
getOriginHost()	String	OriginHost	The fully qualified domain name of the host orginating the CCA reply.
getOriginRealm()	String	OriginRealm	The fully qualified domain name of the realm where the OCS resides that originated the CCA reply.
getAuthApplicationId()	Integer	AuthApplicationId	A value of 4 for Credit Control Applicaton
getCCRequestType()	Integer	CCRequestType	This Enumeration contains the following: • Initial Request • Update Request • Termination Request • Event Request Note: This value will be overridden if using a non-raw method.
getCCRequestNumber()	Integer	CCRequestNumber	Indicates the numbered request within a session. This should be set to 0 for Request Types of Initial and event. It should be incremented by one for each subsequent request within a session.
getCCSessionFailover()	Integer	CCSessionFailover	Indicates whether or not the OCS Supports failover to an alternative Server K \keeping an active session alive. Note, if configured properly, failover will happen automatically through the Diameter Enabler base for a specific destination realm. 1. Failover not supported (This is the default.) 2. Failover supported.
getMultipleServicesCreditControl()	MultipleServicesCreditControl	MultipleServicesCreditControl	Contains all fields for quota management.
getCostInformation()	CostInformation	CostInformation	Contains the cost of a service that the CTF can pass back to the subscriber.

Method Name	Type	Returns	Example values/comments
getCreditControlFailureHandling()	Integer	CreditControlFailureHandling	Holds information telling the CTF what action to take when a transaction times out or some other outage prevents the CTF from completing update transactions. 1. Terminate - Immediately terminate the service. 2. Continue - Attempt to resend the request and continue providing the service for some period of time. 3. Retry and Terminate - Attempt to resend the request if an alternative path is available, terminate the service.

Class ServiceInformation (Ro)

This class is a member of RoChargingInfo and acts as a container for service-specific 3GPP accounting information.

Usage

The 3GPP has separated its vendor-specific AVPs into multiple service-specific charging groups. Contained within ServiceInformation are nested classes that act as holders for service-specific accounting data.

All of the accounting data created by an IMS node is generally located in IMSInformation. However, in some instances, accounting data will need to be generated for multiple services. The Web service client developer simply constructs and sets the service-specific information objects needed to generate accounting information, leaving the irrelevant service objects null.

Within each service-specific class are additional members that represent accounting data that can be processed by an Online Charging System (OCS) to produce CDRs. The type of data recorded will differ across implementations to meet the needs of the accounting application. All figures and statistics must be in a format that complies with the capabilities of the OCS. In the following example, accounting data is created for the SUBSCRIBE event type.

For detailed information regarding the nesting of IMS accounting data, refer to the Javadoc and the 3GPP specifications.

Example

```
RoChargingInfo ccaRequest = new RoChargingInfo();
acctInfo.setSessionId("FQDNServerName:servicexyz:session00001");
acctInfo.setDestinationRealm("userRealm");
acctInfo.setAuthApplicationId(new Integer(4));
acctInfo.setServiceContextId(32260@gpp.org);<codeblock></codeblock><section></section>
ServiceInformation svcInfo = new ServiceInformation();
IMSInformation imsInfo = new IMSInformation();
EventType eventType = new EventType();
eventType.setSipMethod("SUBSCRIBE");
imsInfo.setNodeFunctionality(NodeFunctionality.AS);
imsInfo.setEventType(eventType);

svcInfo.setImsInformation(imsInfo);
acctInfo.setServiceInformation(svcInfo);
service.sendCCInitial(ccaRequest);
```

Get methods

Method	Type	Return	Description
getImsInformation	IMSIformation	imsInformation	Accounting information generated by an IP Multimedia Subsystem service.
getPsInformation	PSInformation	psInformation	Accounting information generated by a Packet Switched service.
getLcsInformation	LCSInformation	lcsInformation	Accounting information generated by a Location Services service.
getMmsInformation	MMSInformation	mmsInformation	Accounting information generated by a Multimedia Messaging service.
getWlanInformation	WLANInformation	wlanInformation	Accounting information generated by a WLAN service.
getPocInformation	PoCInformation	pocInformation	Accounting information generated by a Push-To-Talk Over Cellular service.
getMbmsInformation	MBMSInformation	mbmsInformation	Accounting information generated by a Multimedia Broadcast and Multicast service.

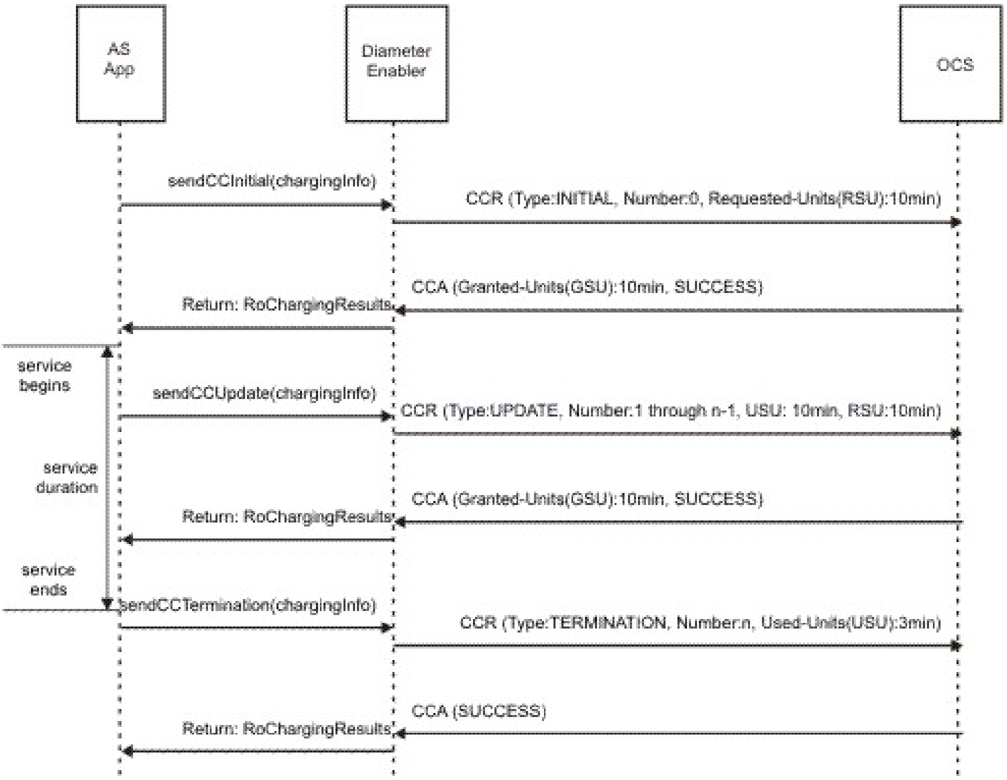
Set methods

Method	Parameter	Type	Description
setImsInformation	imsInformation	IMSInformation	Accounting information generated by an IP Multimedia Subsystem service.
setPsInformation	psInformation	PSInformation	Accounting information generated by a Packet Switched service.
setLcsInformation	lcsInformation	LCSInformation	Accounting information generated by a Location Services service.
setMmsInformation	mmsInformation	MMSInformation	Accounting information generated by a Multimedia Messaging service.
setWlanInformation	wlanInformation	WLANInformation	Accounting information generated by a WLAN service.
setPocInformation	pocInformation	PoCInformation	Accounting information generated by a Push-To-Talk Over Cellular service.
setMbmsInformation	mbmsInformation	MBMSInformation	Accounting information generated by a Multimedia Broadcast and Multicast service.

Ro High-Level API Integrated Examples

The Integrated examples demonstrate the use of the high-level Ro API.

A typical Ro online charging session requires the use of a sendCCInitial() request, one or more sendCCUpdate() requests, and a sendCCTermination() request. These integrated examples demonstrate the use of the API and not the actual architecture of the code that invokes these methods. The application that is making use of these must have a simple state machine to periodically update the Online Charging System (OCS) with quota used and with quota requested to continue providing the service.


AS App
Diameter Enabler
OCS
sendCCInitial(chargingInfo)
CCR (Type:INITIAL, Number:0, Requested-Units(RSU):10min)
CCA (Granted-Units(GSU):10min, SUCCESS)
Return: RoChargingResults
service begins
sendCCUpdate(chargingInfo)
CCR (Type:UPDATE, Number:1 through n-1, USU: 10min, RSU:10min)
service duration
CCA (Granted-Units(GSU):10min, SUCCESS)
Return: RoChargingResults
service ends
sendCCTermination(chargingInfo)
CCR (Type:TERMINATION, Number:n, Used-Units(USU):3min)
CCA (SUCCESS)
Return: RoChargingResults

When in the middle of an online charging session, there may come a time when the OCS needs to interrupt a service to instruct the application providing that service to immediately reauthorize the usage of that service. This is performed by a notifyCCReAuth() callback method provided by the application and invoked by the Diameter Enabler when it receives a Reauthorization Request.

Initiate Credit Control Session with sendCCInitial

This example shows how to initiate a credit control session with sendCCInitial.

In the following snippet, the client application begins a credit control session by constructing an RoChargingInfo object and invoking sendCCInitial. The client requests 60 seconds in Time quota for the subscription identified as ″id0001.″

Example

```
RoChargingInfo roChargingInfo = new RoChargingInfo();

  //Set the Session-Id to a unique identifier.
  roChargingInfo.setSessionId("FQDNServerName:servicexyz:session00001");
  //Set the Destination-Realm of the OCS, which should match a route name in the Diameter_Ro.properties file.
  roChargingInfo.setDestinationRealm("emulators.example.com");
  //Set the CC-Request-Number to 0 for an initial request.
  roChargingInfo.setCcRequestNumber(0);
  roChargingInfo.setServiceContextId("12345@example.com");

  //Set the subscription id to identify the end user's subscription with the OCS.
  SubscriptionId subId = new SubscriptionId();
  subId.setSubscriptionIdData("id0001");
  subId.setSubscriptionIdType(SubscriptionIdType.END_USER_IMSI);
  roChargingInfo.setSubscriptionId(subId);

  MultipleServicesCreditControl mscc = new MultipleServicesCreditControl();

  //Request quota in time Units
  RequestedServiceUnit rsu = new RequestedServiceUnit();
  rsu.setCcTime(60);
  mscc.setRequestedServiceUnit(rsu);
  roChargingInfo.setMultipleServicesCreditControl(mscc);

  RoChargingResults roChargingResults = svc.sendCCInitial(roChargingInfo);
```

Retrieve RoChargingResults for sendCCInitial

This example shows how to retrieve the RoChargingResults object returned by the sendCCInitial.

In the following snippet, the client application retrieves the RoChargingResults object returned by the sendCCInitial method and examines key pieces of information. In this example, the RoChargingResults object indicates that the Online Charging System (OCS) has granted 60 seconds of Time Quota at the cost of 3 US dollars per minute. The client can request 30 more seconds of Time Quota before reauthorization is required.

Example

```
RoChargingResults roChargingResults = svc.sendCCInitial(roChargingInfo);

  MultipleServicesCreditControl mscc = roResult.getMultipleServicesCreditControl();

  //The client can request 30 more seconds of Time Quota before Reauthorization is required.
  Integer timeQuotaThresh = mscc.getTimeQuotaThreshold(); //Example value = 30.

  //The client has been granted 60 seconds of Time Quota.
  GrantedServiceUnit gsu = mscc.getGrantedServiceUnit();
  Integer grantedTimeUnits = gsu.getCcTime();  //Example value = 60

  //Unit cost is 3 US Dollars per minute
  CostInformation costInfo = roResult.getCostInformation();
  String costUnit = costInfo.getCostUnit();  //Example value = "Access to this service is 3.00 a minute."
  Integer currencyCode = costInfo.getCurrencyCode();  //Example Value = 840 (USD)
  UnitValue unitValue = costInfo.getUnitValue();
  Integer exponent = unitValue.getExponent();  //Example value = 1
  Long valueDigits = unitValue.getValueDigits(); //Example value = 3

  //The application uses the retrieved information to manage the credit control session.
  processRoChargingResults(...);
```

Update Credit Control Session with sendCCUpdate

This example shows how to update a credit control session by invoking sendCCUpdate.

In the following snippet, the client application updates a credit control session by constructing a RoChargingInfo object and invoking sendCCUpdate. It is assumed that the client has processed the credit control session information returned in RoChargingResults, as an outcome of the sendCCInitial operation. In this example, the client requests 30 additional seconds of Time quota and reports that 25 seconds of Time quota has been used since the initial request.

Example

```
RoChargingInfo roChargingInfo = new RoChargingInfo();

  //Set the Session-Id to a unique identifier.
  roChargingInfo.setSessionId("FQDNServerName:servicexyz:session00001");
  //Set the Destination-Realm of the OCS, which should match a route name in the Diameter_Ro.properties file.
  roChargingInfo.setDestinationRealm("emulators.example.com");
  //Increment the CC-Request-Number for each additional update request.
  roChargingInfo.setCcRequestNumber(1);
  roChargingInfo.setServiceContextId("12345@example.com");

  SubscriptionId subId = new SubscriptionId();
  subId.setSubscriptionIdData("id0001");
  subId.setSubscriptionIdType(SubscriptionIdType.END_USER_IMSI);
  roChargingInfo.setSubscriptionId("12345@example.com);

  MultipleServicesCreditControl mscc = new MultipleServicesCreditControl();

  //Request additional quota in time units.
  RequestedServiceUnit rsu = new RequestedServiceUnit();
  rsu.setCcTime(30);
  mscc.setRequestedServiceUnit(rsu);

  //Report used quota in time units.
  UsedServiceUnit usu = new UsedServiceUnit();
  usu.setCcTime(25);
  mscc.setUsedServiceUnits(new UsedServiceUnit[]{usu});
  roChargingInfo.setMultipleServicesCreditControl(mscc);

  RoChargingResults roChargingResults = svc.sendCCUpdate(roChargingInfo);
```

Receive Graceful Service Termination - Redirect

This example shows the receiving of a graceful service termination request to redirect to a top-up server

In the following snippet, the client application retrieves the RoChargingResults object returned by the sendCCUpdate method. The client examines key pieces of information, such as the GrantedServiceUnit and CostInformation. In this example, the Online Charging System (OCS) also returns a Final-Unit-Indication, indicating that the final allocation of quota has transpired. After the final used units are reported via a sendCCUpdate, the client is to be redirected to a top-up server to purchase more quota.

Example

```
RoChargingResults roChargingResults = svc.sendCCInitial(roChargingInfo);

  MultipleServicesCreditControl mscc = roResult.getMultipleServicesCreditControl();

  GrantedServiceUnit gsu = mscc.getGrantedServiceUnit();
  Integer grantedTimeUnits = gsu.getCcTime();

  CostInformation costInfo = roResult.getCostInformation();

  String costUnit = costInfo.getCostUnit();
  Integer currencyCode = costInfo.getCurrencyCode();

  UnitValue unitValue = costInfo.getUnitValue();
  Integer exponent = unitValue.getExponent();
  Long valueDigits = unitValue.getValueDigits();

  FinalUnitIndication finalInd = roResult.getMultipleServicesCreditControl().getFinalUnitIndication();
  Integer finalAction = finalInd.getFinalUnitAction();  //Example value = REDIRECT
  RedirectServer redirServer = finalInd.getRedirectServer();
  Integer addressType = redirServer.getRedirectAddressType();  //Example Value = 0 (IPV4_ADDRESS)
  String address = redirServer.getRedirectServerAddress(); //Example Value = An IPV4 address of a top-up server.

  //The client calls sendCCUpdate to report the final used service units, incrementing the CC-Request-Number by 1.
  //Use the redirect server to purchase an additional minute of Time Quota.
```

Retrieve RoReAuthInfo from notifyCCReAuth

This example shows how to retrieve a server-initiated reauthorization request from the Online Charging System (OCS) in the form of a RoReAuthInfo object.

In the following snippet, the client application receives a RoReAuthInfo object from the OCS, requesting reauthorization to continue using the service. The client can query information about the credit pool, service identifier, and rating group.

Example

```
public void notifyCCReAuth(RoReAuthInfo reAuthInfo) {

   GSUPoolReference gsuPoolReference = reAuthInfo.getGSUPoolReference();
   Integer serviceId = reAuthInfo.getServiceIdentifier();
   Integer ratingGroup = reAuthInfo.getRatingGroup();}
```

Reauthorize with OCS

This example shows how to initiate credit reauthorization with sendCCUpdate.

In another example (Receive Graceful Service Termination - Redirect), the client was redirected to a top-up server to purchase more Time Quota. In the following snippet, the client application initiates the credit reauthorization by invoking sendCCUpdate. In this example, there are no UsedServiceUnit to report and the client requests the newly purchased minute of Time Quota.

Example

```
RoChargingInfo roChargingInfo = new RoChargingInfo();

  //Set the Session-Id to a unique identifier.
  roChargingInfo.setSessionId("FQDNServerName:servicexyz:session00001");
  //Set the Destination-Realm of the OCS, which should match a route name in the Diameter_Ro.properties file.
  roChargingInfo.setDestinationRealm("emulators.example.com");
  //Increment the CC-Request-Number for each additional update request.
  roChargingInfo.setCcRequestNumber(3);
  roChargingInfo.setServiceContextId("12345@example.com");

  SubscriptionId subId = new SubscriptionId();
  subId.setSubscriptionIdData("id0001");
  subId.setSubscriptionIdType(SubscriptionIdType.END_USER_IMSI);
  roChargingInfo.setSubscriptionId("12345@example.com);

  MultipleServicesCreditControl mscc = new MultipleServicesCreditControl();

  //Request additional quota in time units.
  RequestedServiceUnit rsu = new RequestedServiceUnit();
  rsu.setCcTime(60);
  mscc.setRequestedServiceUnit(rsu);

  RoChargingResults roChargingResults = svc.sendCCUpdate(roChargingInfo);
```

Receive Graceful Service Termination Request - Terminate

This example shows the receiving of a graceful service termination request to terminate the credit control session.

In the following snippet, the client application retrieves the RoChargingResults object returned by the sendCCUpdate method. The client examines key pieces of information, such as the GrantedServiceUnit and CostInformation. In this example, the Online Charging System (OCS) returns a Final-Unit-Indication, indicating that the final allocation of quota has been issued in the Granted-Service-Units. The final unit action indicates that the client must terminate the charging session after the granted service units have been used

Example

```
RoChargingResults roChargingResults = svc.sendCCUpdate(roChargingInfo);

  MultipleServicesCreditControl mscc = roResult.getMultipleServicesCreditControl();

  GrantedServiceUnit gsu = mscc.getGrantedServiceUnit();
  Integer grantedTimeUnits = gsu.getCcTime();

  CostInformation costInfo = roResult.getCostInformation();

  String costUnit = costInfo.getCostUnit();
  Integer currencyCode = costInfo.getCurrencyCode();

  UnitValue unitValue = costInfo.getUnitValue();
  Integer exponent = unitValue.getExponent();
  Long valueDigits = unitValue.getValueDigits();

  FinalUnitIndication finalInd = roResult.getMultipleServicesCreditControl().getFinalUnitIndication();
  finalInd.getFinalUnitAction();  //Example value = TERMINATE
```

Terminate Credit Control Session with sendCCTermination

This example shows how to terminate a credit control session.

In the following snippet, the client application initiates a graceful termination by invoking sendCCTermination. In this example, the client reports the UsedServiceUnit messages received since the last interim and includes an Event-Timestamp that contains time of session termination.

Example

```
//Set the Session-Id to a unique identifier.
  roChargingInfo.setSessionId("FQDNServerName:servicexyz:session00001");
  //Set the Destination-Realm of the OCS, which should match a route name in the Diameter_Ro.properties file.
  roChargingInfo.setDestinationRealm("emulators.example.com");
  //Increment the CC-Request-Number by 1 for the final termination request.
  roChargingInfo.setCcRequestNumber(2);
  roChargingInfo.setServiceContextId("12345@example.com");
  roChargingInfo.setEventTimeStamp(System.currentTimeMillis());

  SubscriptionId subId = new SubscriptionId();
  subId.setSubscriptionIdData("id0001");
  subId.setSubscriptionIdType(SubscriptionIdType.END_USER_IMSI);
  roChargingInfo.setSubscriptionId(subId);

  MultipleServicesCreditControl mscc = new MultipleServicesCreditControl();

  //Report final used quota in time units.
  UsedServiceUnit usu = new UsedServiceUnit();
  usu.setCcTime(60);
  mscc.setUsedServiceUnits(new UsedServiceUnit[]{usu});
  roChargingInfo.setMultipleServicesCreditControl(mscc);

  RoChargingResults roChargingResults = svc.sendCCTermination(roChargingInfo);
```

Ro Raw API

The Ro Raw API is a more flexible method to send custom session and event requests that require more knowledge of the underlying protocol.

The high-level Ro Web service methods are intended to simplify the programming requirements to create and issue online charging requests. They follow the

Augmented Mackus-Naur Form (ABNF) structures defined by the 3GPP and Internet Engineering Task Force (IETF). However, there are times when this ABNF is not valid, including:

- When new versions of the 3GPP or IETF specifications are available and the new ABNF does not match the old one.
- When an application needs to supplement the ABNF using Attribute Value Pairs (AVPs) that are not defined by the 3GPP or IETF.

When these changes are required, then the high-level Ro Web service methods will not be usable to meet the new ABNF needs. Instead, the sendCCRaw() Web service method must be used.

Usage

The sendCCRaw() Web service allows the developer using this method to completely control the order, type, and content of each AVP to be included in the Diameter Credit Control Request (CCR). These AVPs can be defined by the IETF, the 3GPP, or by any other vendor. The difference is that the application server application must construct an array of these AVPs before invoking the sendCCRaw() method.

When the sendCCRaw() Web service returns, the value returned is an array of the AVPs received from the Online Charging System (OCS) in the Credit Control Answer (CCA) packet. It is then up to the Application Server Application to parse the AVP array and retrieve any of the information that it requires.

The sendCCRaw() Web service is largely a pass-through method. It encapsulates the array of AVPs in the request and places them in a Diameter CCR frame. When the CCA is received, it extracts the AVPs from the CCA, places them into an array, and returns them to the caller. There is very little checking performed by the sendCCRaw() Web service.

AVPs

The IBM WebSphere Diameter Enabler Component communicates with an Online Charging System (OCS) through the Diameter Protocol. Each Diameter Packet contains a list of AVPs, which are used to transfer information between Diameter peers. The Attribute determines the type of information while the Value determines its contents. The IETF and 3GPP have defined a number of AVPs to be used to manage a credit control session.

Note: The format of an AVP header is defined in RFC 3588.

Contained within the AVP header, the AVP Code in combination with the Vendor-ID (if present) uniquely identifies the AVP. AVPs defined in the base specification (RFC 3588) do not contain Vendor-IDs and AVPs defined by the 3GPP use a Vendor-ID of 10415. The AVP flags (VMPrrrrr) further classify the AVP. If the V(endor-Specific) bit is set, the AVP is defined by a vendor other than the IETF. If the M(andatory) bit is set, the AVP MUST be understood by the Diameter peer. The P bit is an obsolete security bit, and should always be zero.

Using Helper Classes

There are several classes provided with the Toolkit to assist you in developing applications that use the Raw interface. The two main jars that hold these classes are:

- DHADiameterPacket.jar
- DHADiameterChargingUtil.jar

DHADiameterPacket.jar contains the classes that make up an AVP. You use the com.ibm.diameter.packet.Avp class when the AVP you wish to send is a base AVP (defined by the IETF). You use the com.ibm.diameter.packet.VsAvp class when the AVP that you wish to send is a vendor-specific AVP. (AVPs defined by the 3GPP are vendor-specific AVPs.)

Note: The internal data or value that the AVP contains is strongly typed. The typing is defined for each AVP. To handle different internal data types, the Diameter Helper classes use an abstract type of com.ibm.diameter.packet.AvpValueUtil. The actual types extend AvpValueUtil and are data types that match those defined by the Diameter specifications. They include:

- AvpValueUtilGrouped
- AvpValueUtilOctetString
- AvpValueUtilUnknown
- AvpValueUtilUnsigned32
- AvpValueUtilUnsigned64
- AvpValueUtilUTF8String

These are fairly self explanatory based on their names. For example, the AvpValueUtilUnsigned32 contains a 32 bit integer. However, the actual value in java is signed, so you must exercise care in using the value especially if you are performing arithmetic with it. You would need to take the hex value and convert it to a long through a logic operations such as an ’AND’.

There are a two AVP types that need some elaboration. The first is the Grouped AVP. Any time the AVP you are using is of type Grouped, it contains other AVPs. The AvpValueUtilGrouped helper class contains a vector of AVPs.

The other AVP type is AvpValueUtilUnknown. This AVP type is used by the Diameter Enabler when it is attempting to parse the data stream, but does not know the definition for a given AVP. A case when this might happen is when a server defines its own, custom Vendor Specific AVP outside of the IETF or 3GPP. The Diameter Enabler, when it receives an AVP that it does not recognize will store all of the data for that AVP as a byte array or octet string. So, if the custom AVP was of type unsigned 32, then the Diameter Enabler would read it in to array with a length of 4 bytes and place this value in an AvpValueUtilUnknown object in the AVP.

Thus, when you are using the Raw interface to create a custom frame and expect to receive back custom AVPs. The AVPs unknown to the Diameter Enabler will have AvpValueUtilUnknown values in them. The application must then convert that byte array to the appropriate type.

Any AVP that is unknown to the Diameter Enabler will receive this handling including custom Grouped AVPs. Even if some of the AVPs that a Grouped AVP holds are known to the Diameter Enabler, they will be included as data in the byte array that contains the payload of the owning Grouped AVP.

For more information on the helper classes and their methods, refer to the Javadoc. These classes are provided as part of the IMS Connector CD. They are also included in the IMS Tooling packages associated with the IMS Connector.

Defining Your Own AVPs

You can define any AVP that you wish to send in the CCR message. Most of the AVPs defined in the IETF RFCs 3588 and 4006 are already defined within the Diameter Enabler helper classes. You can simply create an AVP by providing the data to an AvpFactory method. For example:

```
Avp originRealmAvp = AvpFactory.createOriginRealmAvp("asclient.example.com");
```

This example creates an Origin-Realm AVP whose value is asclient.example.com. Many of these AVPs are vendor specific for IMS with a vendor ID of 10415. Here is an example of the way in which ChargingAvpFactory is used:

```
Avp exponentAvp = ChargingAvpFactory.createExponentAvp(2);
```

The AVPs that you are able to create using the AvpFactory and ChargingAvpFactory are limited to those defined by the Diameter Enabler. You may want to create your own AVPs that are not included in the Diameter Enabler.

You can do this using the helper classes: com.ibm.diameter.packet.Avp and com.ibm.diameter.packet.AvpValueUtil. For example:

```
AvpValueUtil numberOfMembersValueUtil = new AvpValueUtilUnsigned32(20);
Avp numberOfMembersAvp = new VsAvp(NUMBER_OF_MEMBERS,
     Avp.VON_MON_POFF,numberOfMembersValueUtil, VENDOR_ID_COMPANYX);
```

This example creates a new vendor-specific AVP of type Unsigned32 and loads it with a value of 20. It also creates the AVP with the V-bit (vendor -specific) set to 1, the M-bit (Mandatory) set to 1, and the P-bit set to 0. Finally, the Vendor ID is set to COMPANYX's vendor ID. The vendor ID should be listed in the enterprise-numbers2 on the Internet Assigned Numbers Authority site.

Receiving and Processing the Results of Your Raw Request

You create AVPs when you wish to send a data element to the OCS. You receive back the AVPs that the OCS sends in its CCA. The result is returned in an Array of AVPs that occur in the order in which the AVPs were received with array[0] being the first AVP received. The type of data that is held within the AVP is either that defined by the IETF or 3GPP if the AVP is formally defined in the Ro definition. If the AVP is not in any of these definitions, then the data type will be AvpValueUtilUnknown.

Whenever you receive an AVP with data of type AvpValueUtilUnknown, the Diameter Enabler does not have a definition of the AVP in its dictionary. Your application must convert the byte array into data of a type that is usable by your application. Simple types, such as Unsigned32, are straight forward. However, the Grouped type means that AVPs including their AVP headers will be included in the byte array. And because there can be any number of nesting levels, Grouped AVPs will need some additional parsing algorithm to convert them from an array of bytes into an array of AVPs or other data elements.

In general, if you wish to retrieve information from the reply frame, you will walk through the list of AVPs received, pull the one of interest, get its AvpValueUtil, and invoke the getAvpValue() method of the AvpValueUtil.

Limitations of the Raw Interface

If you create an AVP that is already defined by the Diameter Enabler, your AVP will be transmitted with the type, values, and AVP header bits that you have defined. However, if the same AVP is received by the Diameter Enabler in the reply, the Diameter Enabler’s definition will be applied when creating the AVP.

Also, an incorrectly created AVP may affect the data stream and cause an Out-Of-Sync error to occur. If when transmitting, the number of bytes expected does not match the length parameters, the packet will be malformed. An error such as this may cause the other side of the connection to determine that it no longer knows where it is in the data stream. This generally results in the connection being dropped and re-established. For this reason, you need to secure and limit the access to these data services to trusted applications.

Example

```
DiameterRoService_SEIServiceLocator locator = new DiameterRoService_SEIServiceLocator();
DiameterRoService_SEI service = locator.getDiameterRoService(endpoint);

Avp[] avpArray = null;
ArrayList avpArrayList = new ArrayList();
avpArrayList.add(AvpFactory.createSessionIdAvp("session1"));
avpArrayList.add(AvpFactory.createOriginHostAvp("aaa://originhost.example.com;transport=tcp"));
avpArrayList.add(AvpFactory.createOriginRealmAvp("aaa://example.com;transport=tcp"));
avpArrayList.add(AvpFactory.createDestinationRealmAvp("example.example.com"));
avpArrayList.add(AvpFactory.createDestinationHostAvp("aaa://host1.example.example.com;transport=tcp"));
avpArrayList.add(AvpFactory.createAuthApplicationIdAvp(4)); // credit control application

avpArrayList.add(createVSApplicationIdAvp()); // Grouped

avpArrayList.add(ChargingAvpFactory.createServiceContextIdAvp("Video Share is vshare@example.com"));
avpArrayList.add(AvpFactory.createEventTimestampAvp(System.currentTimeMillis()));
avpArrayList.add(ChargingAvpFactory.createCcRequestTypeAvp(1)); // 1 = Initial
avpArrayList.add(ChargingAvpFactory.createCcRequestNumberAvp(0)); // Initial CCR number = 0

avpArrayList.add(createSubscriptionIdAvp()); // Grouped

avpArrayList.add(createImsInformationAvp()); // Grouped

avpArray = avpArrayList.toArray(avpArray);

//Send the Web Service Request
Avp[] result = service.sendCCRaw(avpArray);
```

Sh subscriber profile Web service

An IMS Application Server uses Sh subscriber profile Web service to get and update user profile data from the Home Subscriber Server (HSS).

In addition to getting and updating data, the Sh subscriber profile Web service includes methods for subscriptions and notifications to receive updates when information in the HSS changes.

Transaction types

The Sh subscriber profile Web service is the interface between the IMS Application Server or an Open Service Access - Service Capability Server (OSA-SCS) and the HSS. The Sh subscriber profile Web service provides the IMS Application Server with secure access to subscriber data centrally stored in the HSS.

Sh subscriber profile Web service provides interfaces for:

- Data handling operations for getting data from the HSS and updating repository data to the HSS
- Subscription and notification operations for an IMS Application Server to subscribe to receive notification from the HSS when data changes and for the HSS to notify an IMS Application Server when data that is subscribed to changes

User data types

The Sh application operates on user profiles. A user profile is stored in the HSS. These profiles enable the user to perform authentication and authorization of the user.

A user profile is bound to a Private User Identity. The Sh application uses the term user data to refer to diverse data types in a user profile. These data types are defined as an XML schema, `ShDataType.xsd`, attached to the ETSI TS 129.328 standards. The user data can refer to any of the following:

- Repository Data: This is transparent data stored on the HSS.
- IMS Public Identity: A list of Public User Identities allocated to each user.
- IMS User State: This is the registry state of the user in IMS.
- S-CSCF: This contains the address of the serving- call session control function (S-CSCF)
- Initial Filter Criteria: Contains the triggering information that indicates the appropriate service to be called.
- Location Information: This contains the location of a user in the packet-switched or circuit-switched domains.
- User State: This contains the state of the user in the packet-switched or circuit-switched domain.
- Charging information: This contains the address of the charging functions.
- PSI Activation: This contains the state of a Public Service Identity (PSI) indicating whether or not it is active.

Data operations

The Sh subscriber profile Web service application implements four Diameter Sh data operations that enable the exchange of user data between an application and the Home Subscriber Server. An application interfaces with the Diameter Enabler base through the Sh subscriber profile Web service.

The following table lists the four types of Diameter Sh data operations, with their appropriate commands. The Diameter Sh protocol defines new AVPs and eight new Diameter commands to support user data operations.

Table 9. Data operations

Data operation	Protocol description
Data get	Allows an application server to get data from the HSS for a particular user. The user data can be of any of these: • Repository Data: This is transparent data stored on the HSS. • IMS Public Identity: A list of Public User Identities allocated to each user. • IMS User State: This is the registry state of the user in IMS. • S-CSCF: This contains the address of the serving- call session control function (S-CSCF) • Initial Filter Criteria: Contains the triggering information that indicates the appropriate service to be called. • Location Information: This contains the location of a user in the packet-switched or circuit-switched domains. • User State: This contains the state of the user in the packet-switched or circuit-switched domain. • Charging information: This contains the address of the charging functions topic. • MSISDN: This contains the user's MSISDN. • PSI Activation: This contains the state of the activation. • User Data: This contains user profile data in an XML string representation.
Data update	Allows an application to update repository data and PSI activation for a specific user and store them in the HSS.
Subscribe	Allows an application to subscribe to notification when the data has been updated in the HSS. Notification can be sent for the following data types: • Repository data • IMS user state • S-CSCF name • Initial filter criteria • PSI Activation value • User data
Notify	Allows an HSS to notify the application that the data it subscribed for notification have been updated. The data are then sent to the application server by the HSS in the notification. Notification can be sent for the following data types: • Repository data • IMS user state • S-CSCF name • Initial filter criteria • PSI Activation value • User data

Sh subscriber profile Web service call flows

Call flows describe how the applications get and update data, as well as how applications subscribe to data change notification, and how notifications about user state are passed between the IMS Application Server, Diameter Enabler base, and the Home Subscriber Server (HSS).

Data get (User Data Request) flow:

1. IMS Application Server invokes one of the Sh data get Web service requests, such as getRepositoryData.
2. The Sh subscriber profile Web service application validates the parameters received from the Web service interface and then builds a Sh User Data Request message using the parameters. Diameter Enabler base then sends the User Data Request message to the HSS.
3. The HSS receives the User Data Request message, retrieves the requested data, and then returns the data to the Diameter Enabler base in a User Data Answer message.
4. The Sh subscriber profile Web service application receives the User Data Answer message from the Diameter Enabler base containing the requested data in XML format. The Sh subscriber profile Web service application parses the XML data received and returns the information to the Web service interface.

Data update (Profile Update Request):

1. IMS Application Server invokes one of the Sh data update Web service requests, such as updateRepositoryData.
2. The Sh subscriber profile Web service application validates the parameters received from the Web service interface and then builds a Sh Profile Update Request message using the parameters. Diameter Enabler base then sends the Profile Update Request message to the HSS.
3. The HSS receives the Profile Update Request message, updates the requested profile information, and returns the result to the Diameter Enabler base in a Profile Update Answer message.
4. The Sh subscriber profile Web service application receives the Profile Update Answer message from the Diameter Enabler base that contains the result of the profile update operation. The Sh subscriber profile Web service application returns the result to the Web service interface.

Data subscribe (Subscribe Notifications Request):

1. IMS Application Server invokes one of the Sh data subscription Web service requests, such as subscribeRepositoryData.
2. The Sh subscriber profile Web service application validates the parameters received from the Web service interface and then builds a Sh Subscription Notifications Request message using the parameters. Diameter Enabler base then sends the Subscription Notifications Request message to the HSS.
3. The HSS receives the Subscription Notifications Request message, creates a subscription to the requested profile information, and returns the result to the Diameter Enabler base in a Subscription Notifications Answer message.
4. The Sh subscriber profile Web service application receives the Subscription Notification Answer message from the Diameter Enabler base that contains the result of the Subscribe Notification operation and returns the result to the Web service interface.

Data notify (Notification Request):

1. To receive notifications the IMS Application Server must first implement the server side Notify WSDL and register the Web service endpoint as the callback URI in all subscriptions.
2. When data with an associated subscription is updated in the HSS, the HSS sends a Push Notification message including the updates to the Diameter Enabler base.
3. The Diameter Enabler base sends the message to the Sh subscriber profile Web service application. The Sh subscriber profile Web service application references the subscriptions associated with the notification in the database to retrieve the callback URI.
4. The Sh subscriber profile Web service application sends the updated data received in the notification request to the appropriate notification Web services.

Sh subscriber profile Web service

The Sh subscriber profile Web service provides subscriber profile services.

The interface is comprised of a number of methods.

Get Web service methods

The get Web service methods are used when an IMS Application Server retrieves user profile data from the HSS.

When using the basic Web services for the Sh operation, you should note the following nuances in naming conventions. A data type, which consists of one or more joined words with a capital letter starting each word, is equivalent to the corresponding XML tag in the `ShDataType.xsd`. For example: data type *RepositoryData* is equivalent to the XML tag RepositoryData in `ShDataType.xsd`.

Nine types of user data can be accessed using the get Web service methods. These are required for compliance with the Sh specifications. The Sh subscriber profile Web service are grouped together into two categories based on the data types of parameters and return. An application client chooses one to use for a particular operation depending on its preference. These three categories are:

- Simple type: these Web services take as parameter and/or return simple data types. These data types are either Java data types or defined in the `ShDataType.xsd`.
- XML type: these Web services take as parameter or return a user profile XML data string. The user profile XML is defined in `ShDataType.xsd`.

getRepositoryData Web service method:

This Web service method returns the RepositoryData containing transparent data for a particular subscriber. This data is only understood by the IMS Application Server that implements the services.

Usage

The getRepositoryData Web service method is used to access the user profile repository data from the HSS. The format of the RepositoryData is defined in the `ShDataType.xsd` as type TransparentData. Some or all values within the RepositoryData object may be null if the corresponding element data (defined in ShDataType.xsd) was not received by the HSS.

Example

```
RepositoryData rdata = service.getRepositoryData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234@example.com", "IBM-Diameter-SH-1234567");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
serviceIndication	String	A unique identifier for the requested service data. Example: IBM-Diameter-SH-1234567.

getPublicIdentifiers Web service method:

This Web service method returns a PublicIdentifiers object containing Public User Identities, Public Service Identities, or MSISDN numbers, allocated to each IMS subscriber. Multiple identities and MSISDN may be returned.

Usage

The getPublicIdentifiers Web service method returns an array of URI elements specifying an IMS Public User Identity or a Public Service Identity. These will be either a SIP or TEL URI. The getPublicIdentifiers Web service method may also return an array of Mobile Subscriber ISDN Number (MSISDN).

Example

```
PublicIdentifiers myPublicIdentities = service.getPublicIdentifiers( "aaa://host.example.com;protocol=diameter;-117302099;1",
"sipintel110.city.company.com", "realmA.mycompany.com", "sips:carol@ws1234.domain2.com", "0113 272 2245", 0);
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245
identitySet	int	The following values are defined in `ShDataType.xsd`: 0 = ALL_IDENTITIES 1 = REGISTERED_IDENTITIES 2 = IMPLICIT_IDENTITIES

getIMSUserState Web service method:

This Web service method returns the registry states for the user.

Usage

The user state can be registered, unregistered, pending while being authenticated, or unregistered. A service call session control function (S-CSCF) name is allocated to trigger services for unregistered users. The return value is an integer that contains the IMS User State of the public identifier being referenced. The possible values are explained in the following table:

Table 10. Return Values

Integer Value	User State
0	NOT_REGISTERED
1	REGISTERED
2	REGISTERED_UNREG_SERVICES
3	AUTHENTICATION_PENDING

Example

```
myIMSUserStateName = service.getIMSUserState( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:carol@ws1234.domain2.com");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

getSCSCFName Web service method:

This Web service method returns the address of the Serving Call Session Control Function Name (*S-CSCF*) allocated to the user.

Usage

This Web service method returns a URI which identifies a Serving Call Session Control Function Name where a multimedia public identity is recognized.

Example

```
URI scsfName = service.getSCSFName( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel110.city.company.com",
"realmA.mycompany.com", "sips:carol@ws.1234.domain2.com");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

getInitialFilterCriteria Web service method:

This Web service method returns one or more InitialFilterCriteria objects that contain a collection of user-related information that helps a S-CSCF determine when to invoke a particular IMS Application Server to provide a service.

Usage

Filter criteria are used to determine the services that will be provided to the collection of Public User Identities listed in each user's *user_profile*. Filter criteria contains a collection of user-related information that helps the *S-CSCF* decide when to involve a particular IMS Application Server to provide a service. For example: the InitialFilterCriteria object contains a collection of user information that helps the *S-CSCF* decide when to forward the SIP request to a particular application server.

This Web service method returns an array of *InitialFilterCriteria* objects that adhere to the *InitialFilterCriteria* schema defined in the ShDataType.xsd. Multiple filter criteria may be returned from the HSS.

Example

```
InitialFilterCriteria[] initialFilterCriteria = service.getInitialFilterCriteria( "aaa://host.example.com;protocol=diameter;-117302099;1",
"sipintel10.city.company.com", "realmA.mycompany.com", "sips:carol@ws1234.domain2.com", "sip:siphappens.domain.com" );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
serverName	URI	The SIP URL of the application server that is providing the service of interest.

getCSLocationInformation Web service method:

This Web service method returns the location information of a user or service in a circuit-switched domain.

Usage

This Web service method returns a CSLocationInformation object specifying the location information of a user or service in a circuit-switched domain. The following data elements in the CSLocationInformation received from the HSS may be Base64 encoded:

- LocationNumber
- CellGlobalId
- ServiceAreaId
- LocationAreaId
- GeographicalInformation

- GeodeticInformation
- VLRNumber
- MSCNumber

Example

```
CSLocationInformation myCSLocationInformation = service.getCSLocationInformation( "aaa://host.example.com;protocol=diameter;117302099;1",
"sipintel10.city.company.com", "realmA.mycompany.com", "0113-272-2245" );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245

getPSLocationInformation Web service method:

This Web service method will return location information for a user or a service in a packet-switched network. The content of the PSLocationInformation object is defined in SHDataType.xsd. Some or all values within the PSLocationInformation object may be null if the corresponding element data (defined in ShDataType.xsd) was not received from the HSS.

Example

The following data elements in the PSLocationInformation received from the HSS may be Base64 encoded:

- CellGlobalId
- ServiceAreaId
- LocationAreaId
- RoutingAreaId
- GeographicalInformation

- GeodeticInformation
- SGSNNumber

```
PSLocationInformation myPSLocationInformation = service.getPSLocationInformation( "aaa://host.example.com;protocol=diameter;117302099;1",
"sipintel10@city.example.com", "realmA.example.com", "0113 272 2245" );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245

getCSUserState Web service:

This Web service method returns the state of the user in the circuit-switched (CS) domain. If a null CSUserState element is received from the HSS, the value returned will be -1.

Usage

The user state will return an integer value which contains the state of the user in the CS domain. The possible integer values are explained in the following tables:

Table 11. Circuit-Switched (CS) Domain Status

Integer Value	CS Domain Status
0	CAMEL_BUSY
1	NetworkedDeterminedNotReachable
2	AssumeIdle
3	NotProvidedfromVLR

Example

```
intgetCSUserStateName = ( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:carol@ws1234.domain2.com", "0113 272 2245", "2");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245

getPSUserState Web service method:

This Web service method returns the state of the user in the packet-switched (PS) domain. If a null PSUserState element is received from the HSS, the value returned will be -1.

Usage

The user state will return an integer value which contains the state of the user in the PS domain. The possible integer values are explained in the following table:

Table 12. Packet-Switched (PS) Domain Status

Integer Value	PS Domain Status
0	Detached
1	AttachedNotReachableForPaging
2	AttachedReachableForPaging
3	ConnectedNotReachableForPaging
4	ConnectedReachableForPaging
5	NotProvidedFromSGSN

Example

```
int psUserState = service.getPSUserState( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.company.com",
null, "realmA.mycompany.com", "0113 272 2245");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245

getChargingInformation Web service method:

This Web service method returns the AAA (Authentication and Authorization) URIs of the primary and secondary event charging function and CCF.

Usage

This Web service method returns a ChargingInformation object that contains the Diameter URIs of the charging functions.

Example

```
ChargingInformation info = service.getChargingInformation( "aaa://host.example.com;protocol=diameter;-117302099;1",
"sipintel110.city.company.com", "realmA.mycompany.com", "0113 272 2245" );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22 When requesting IMSPublicIdentity or ChargingInformation, either the publicUserIdentity or msisdn parameter is required.

getMSISDN Web service method (Deprecated):

This deprecated Web service method returns an array of MSISDN numbers associated with this IMS subscriber's Public User Identity.

Usage

This deprecated Web service method returns an array of strings which contain the mobile subscriber number. An MSISDN consists of a country code, a national destination code, and a subscriber's number.

Example

```
String[] carolMSISDNs = service.getMSISDN( "aaa://host.example.com;protocol=diameter;117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:carol@ws1234.domain2.com" );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

getMSISDNList Web service method:

This Web service method returns an array of MSISDN numbers associated with this IMS subscriber's Public User Identity or other MSISDN numbers.

Usage

This Web service method returns an array of strings which contain the mobile subscriber number. A MSISDN consists of a country code, a national destination code, and a subscriber's number. A Public User Identity (PUI) may have many MSISDN numbers associated with it. These numbers may be used to establish a connection or session with the subscriber whose MSISDN list is retrieved. The MSISDN list can be retrieved by specifying either the PUI or the MSISDN numbers already associated with the PUI.

Example

Using PUI to retrieve the list:

```
String[] carolMSISDNList = service.getMSISDNList( "aaa://host.example.com;protocol=diameter;117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:carol@ws1234.domain2.com", null );
```

Using MSISDN to retrieve the list:

```
String[] carolMSISDNList = service.getMSISDNList( "aaa://host.example.com;protocol=diameter;117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", null, "+919-123-4567");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245 When requesting IMSPublicIdentity or ChargingInformation either the **public_user_identity** or **msisdn** parameter is required.

getUserData Web service method:

This Web service method returns user profile data in an XML string representation. The XML file adheres to the format defined in the `ShDataType.xsd` schema file.

Usage

This Web service method returns the following types of user profile data as an XML string representation: *Repository Data*, *IMSPublic Identity*, *IMSUserState*, *S_CSCFName*, *InitialFilteCriteria*, *CSLocationInformation*, *PSLocationInformation*, *CSUserState*, *PSUserState*, *Charging_information*, *MSISDN*, and *PSIActivation*.

Example

The getUserData Web service method can be used to request different types of data. The parameters that are required will depend on the type of data being requested.

Note: In the following examples, the sixth parameter is used to retrieve the specific type of User Data from the user's profile.

Example for a request RepositoryData user data XML string:

```
String repositoryDataXML = service.getUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", null, 0, "IBM-Diameter-SH-1234567", -1, -1, null);
```

Example for request InitialFilterCriteria user data XML string:

```
String filterCriteriaXML = service.getUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", null, 13, null, -1, -1, new URI("sip:siphappens.example.com") );
```

Example for request IMSPublicIdentity user data XML string:

```
String publicIdentityXML = service.getUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", null, 10, null, 1, -1, null);
```

Example for request CSLocationInformation user data XML string:

```
String csLocInfoXML = service.getUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", null, "0113 272 2245", 14, null, -1, 0, null);
```

Example for request PSIActivation user data XML string:

```
String psiActivationXML = service.getUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", null, "0113 272 2245", 18, null, -1, null, -1);
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

Parameter Name	Type	Description
msisdn	String	Mobile Subscriber ISDN is the standard international telephone number used to identify a given subscriber. The MSISDN is the telephone number of a GSM (Global System for Mobile Communications) cell phone that is stored in the SIM (Subscriber Identity Module) smart card inside the phone. A public user identity in IMS is the equivalent of the MSISDN in a GSM network. Example: +919-123-4567 (919)-155-4567 0113 272 2245 0044 113 272 2245 +44 113 272 2245 When requesting IMSPublicIdentity or ChargingInformation either the public_user_identity or msisdn parameter is required.
dataReference	int	Data reference values indicate the type of user data being requested: 0 = Repository Data 10 = IMSPublicIdentity 11 = IMSUserState 12 = S_CSCFName 13 = InitialFilterCriteria 14 = LocationInformation 15 = UserState 16 = Charging_Information 17 = MSISDN 18 = PSI Activation
serviceIndication	String	A unique identifier for the requested service data. Example: IBM-Diameter-SH-1234567. This parameter is only required if the dataReference value is set to 0 for the repository data.
identitySet	int	The following values are defined in `ShDataType.xsd`: 0 = ALL_IDENTITIES 1 = REGISTERED_IDENTITIES 2 = IMPLICIT_IDENTITIES This parameter is only required if the dataReference value is set to 10 for the IMSPublicIdentity.
serverName	URI	The SIP URL of the application server that is providing the service of interest. This parameter is only required if the dataReference value is set to 13 for the initialFilterCriteria.
requestedDomain	int	This parameter is required only if the dataReference value is set to 14 or 15 for LocationInformation or UserState. 0 = CS_Domain 1 = PS_Domain

getPSIActivation Web service method:

This Web service method returns a value indicating the current state of activation (active or inactive) of a service provided by an application.

Usage

This Web service method returns a value indicating the current state of activation (active = 1 or inactive = 0) of a service provided by an application. This enables the entire IMS network to be aware whether the service is either available (active) or not (inactive).

Example

```
int carolPSIActivation = service.getPSIActivation( "aaa://host.example.com;protocol=diameter;117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:carol@ws1234.domain2.com" );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

Update Web service methods

The update Web service methods allow an IMS Application Server to update repository data for a specific user and stores that data in the HSS.

updateRepositoryData Web service method:

This Web service method is used by an IMS Application Server to update the transparent data stored in the HSS for a user.

Usage

updateRepositoryData is used to update the user profile repository stored in the HSS. The returned string contains a result code from the HSS.

Example

```
String results = service.updateRepositoryData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:carol@ws1234.domain2.com", "IBM-Diameter-SH-1234567", 36, "service data");
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
serviceIndication	String	A unique identifier for the requested service data. Example: IBM-Diameter-SH-1234567.
sequenceNumber	int	The sequence number of the service data that is to be updated by the HSS. Values must be any integer in the following range: 0 - 65535.
serviceData	String	New data to be updated in the user repository.

updateRepositoryDataByXML Web service method:

This Web service method is used by an IMS Application Server to update the transparent data stored in the HSS for a user.

Usage

This Web service method is used to update the XML string representation of the user profile repository data from the HSS. This format of this data is defined as RepositoryData in `ShDataType.xsd`. The string contains a result code.

Example

Example of the initial creation of user profile information. Notice the sequence number specified is zero.

```
String user_data = "<?xml version=\"1.0\"
encoding=\"UTF-8\"?><Sh-Dataxmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"ShDataType.xsd\"><RepositoryData><ServiceIndication>IBM-Diameter-SH-1234567</ServiceIndication>
<SequenceNumber>0</SequenceNumber><ServiceData><Data>email
```

```
bjjones@example.com *</Data></ServiceData></RepositoryData></Sh-data>";

String resultCode = service.updateRepositoryDataByXML("aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", user_data);
```

Parameters

Parameter Name	Type	Description
sessionID	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
userData	String	The XML string representation of the user profile data which must comply with `ShDataType.xsd`.

updatePSIActivation Web service method:

This Web service method updates the value held by the HSS indicating the current state of activation (active or inactive) of a service provided by an application.

Usage

This Web service updates the value held by the HSS indicating the current state of activation (active or inactive) of a service provided by an application.

Example

```
String applicationPSIActivation = service.updatePSIActivation( "aaa://host.example.com;protocol=diameter;117302099;1", "sipintel10.city.company.com",
"realmA.mycompany.com", "sips:app1@ws1234.domain2.com", 1 );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1

Parameter Name	Type	Description
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
activation	int	The current activation state of a specific service. Example: 0 = Inactive 1 = Active.

Subscribe Web service methods

An IMS Application Server uses the subscribe Web service methods to subscribe to notification when data is updated from the HSS.

A subscription is allowed for the following data types:
- Repository data
- IMS user state
- S-CSCFName
- Initial filter criteria
- PSI activation value

subscribeRepositoryData Web service method:

This Web service method is used for an IMS Application Server to subscribe to notification when user data is updated from the home subscriber server (HSS).

Usage

The subscribeRepositoryData Web service method is used by an IMS Application Server to subscribe or unsubscribe to notifications from the HSS when repository data for the publicIdentity and serviceIndication change. The returned string contains a result code.

Example

Example for Subscribe to RepositoryData updates:

```
Callback URI = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService")
String results = service.subscribeRepositoryData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.example.com" ,
"realmA.example.com", "sips:carol@ws1234.example.com", "IBM-Diameter-SH-1234567", 0,callbackURI , null, null );
```

Example for Unsubscribe to RepositoryData updates:

```
Callback URI = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService")
String results = service.subscribeRepositoryData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.example.com" ,
"realmA.example.com", "sips:carol@ws1234.example.com", "IBM-Diameter-SH-1234567", 1,callbackURI , null, null );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
serviceIndication	String	A unique identifier for the requested service data. Example: IBM-Diameter-SH-1234567.
subsReqType	int	Available subscription status values: 0 = SUBSCRIBE 1 = UNSUBSCRIBE
callbackUri	URI	The Web service URI used to deliver notification back to the subscriber; for example: http://example.com:7676/DHADiameterShNotifyService/services/DiameterShNotifyService
userID	String	If the IMS Application Server Notify Web service is protected, specify the user ID for that Web service.
password	String	If the IMS Application Server Notify Web service is protected, specify the password for that Web service.

subscribeIMSUserState Web service method:

This Web service method subscribes to notifications of changes in the registry states for the public identity specified.

Usage

This Web service method is invoked by an IMS Application Server to subscribe to notification to changes in the registry state for the associated public identity.

Example

```
URI callbackURI = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService")

String resultCode = service.subscribeIMSUserState( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 0, callbackURI, null, null);
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
subsReqType	int	Available subscription status values: 0 = SUBSCRIBE 1 = UNSUBSCRIBE
callbackUri	URI	The Web service URI used to deliver notification back to the subscriber; for example: http://example.com:7676/DHADiameterShNotifyService/services/DiameterShNotifyService
userId	String	If the IMS Application Server Notify Web service is protected, specify the user ID for that Web service.
password	String	If the IMS Application Server Notify Web service is protected, specify the password for that Web service.

subscribeInitialFilterCriteria Web service method:

This Web service method is used by an IMS Application Server to subscribe to notifications of changes in the filter criteria. The notifications are specific to a particular public identity and application server combination.

Usage

This Web service method returns a string which contains the result code of the action.

Example

```
URI callbackUri = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService");

String resultCode = service.subscribeInitialFilterCriteria( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 0, new URI("sip:siphappens.example.com"), callbackUri, null, null);
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
subsReqType	int	Available subscription status values: 0 = SUBSCRIBE 1 = UNSUBSCRIBE
serverName	URI	The SIP URL of the application server that is providing the service of interest.
callbackUri	URI	The Web service URI used to deliver notification back to the subscriber; for example: http://example.com:7676/DHADiameterShNotifyService/services/DiameterShNotifyService
userId	String	If the IMS Application Server Notify Web service is protected, specify the user ID for that Web service.
password	String	If the IMS Application Server Notify Web service is protected, specify the password for that Web service.

subscribeUserData Web service method:

This Web service method is used to subscribe to notifications for the following data_reference types: RepositoryData, InitialFilterCriteria, S-CSCFName, and IMSUserState. The notification data is returned in an XML string representation, that adheres to the schema defined in `ShDataType.xsd`.

Usage

The result code for the subscribe operation is returned.

Example

An example is provided for each type of data because the parameter requirements are different

```
URI callbackUri = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService");
```

Example for Subscribe to RepositoryData notifications:

```
String subscribeResult = service.subscribeUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 0, "IBM-Diameter-SH-1234567", 0, null, callbackUri, null, null);
```

Example for Subscribe to InitialFilterCriteria notifications:

```
URI serverUri = new URI("sip:siphappens.example.com");
String subscribeResult = service.subscribeUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 13, null, 0, serverUri, callbackUri, null, null );
```

Example for Subscribe to S-CSCFName notifications:

```
String subscribeResult = service.subscribeUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 12, null, 0, null, callbackUri, null, null );
```

Example for Subscribe to IMSUserState notifications:

```
String subscribeResult = service.subscribeUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 11, null, 0, null, callbackUri, null, null );
```

Example for Subscribe to PSIActivation notifications:

```
String subscribeResult = service.subscribeUserData( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10@city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 18, null, 0, null, callbackUri, null, null );
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicUserIdentity	String	A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

Parameter Name	Type	Description
dataReference	int	Data reference values indicate the type of user data being requested: 0 = Repository Data 11 = IMSUserState 12 = S_CSCFName 13 = InitialFilterCriteria 18 = PSI Activation
serviceIndication	String	A unique identifier for the requested service data. Example: IBM-Diameter-SH-1234567. This parameter is only required if the dataReference value is set to 0 for the repository data.
subsReqType	int	Available subscription status values: 0 = SUBSCRIBE 1 = UNSUBSCRIBE
serverName	URI	The SIP URL of the application server that is providing the service of interest. This parameter is only required if the dataReference value is set to 13 for the initialFilterCriteria.
callbackUri	URI	The Web service URI used to deliver notification back to the subscriber; for example: http://example.com:7676/DHADiameterShNotifyService/services/DiameterShNotifyService
userID	String	If the IMS Application Server Notify Web service is protected, specify the user ID for that Web service.
password	String	If the IMS Application Server Notify Web service is protected, specify the password for that Web service.

subscribeSCSCFName Web service:

This Web service method is used for an IMS Application Server to subscribe to notification when the S-CSCFName for the associated public identity is updated from the home subscriber server (HSS)

Usage

The IMS Application Server uses this Web service to subscribe or unsubscribe to notifications from the HSS when the S-CSCFName, associated with the publicIdentity, specifies changes.

Example

For subscribeS-SCSFName Web service:

```
URI callbackURI = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService")

String resultCode = service.subscribeSCSCFName( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.example.com",
"realmA.example.com", "sips:carol@ws1234.example.com", 0, CallbackURI, null, null);
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com This is an optional parameter. If specified, it must be the FQDN of destination Diameter node which handles the request.
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
subsReqType	int	Available subscription status values: 0 = SUBSCRIBE 1 = UNSUBSCRIBE
callbackUri	URI	The Web service URI used to deliver notification back to the subscriber; for example: http://example.com:7676/DHADiameterShNotifyService/services/DiameterShNotifyService
userId	String	If the IMS Application Server Notify Web service is protected, specify the user ID for that Web service.
password	String	If the IMS Application Server Notify Web service is protected, specify the password for that Web service.

subscribePSIActivation Web service method:

This Web service method allows the caller to subscribe for changes to the state (Active or Inactive) of a service provided by an application.

Usage

The IMS Application Server uses this Web service to subscribe or unsubscribe to notifications from the HSS when the PSI Activation, associated with a specific Public Service Identity, has changed states.

Example

```
URI callbackURI = new URI("http://hostname.example.com:1234/DHADiameterShNotifyTestClient/services/DiameterShNotifyService");
String resultCode = service.subscribePSIActivation( "aaa://host.example.com;protocol=diameter;-117302099;1", "sipintel10.city.example.com", "realmA.example.com
```

Parameters

Parameter Name	Type	Description
sessionId	String	Identifies a specific session. All messages pertaining to a specific session must include only one Session-Id AVP and the same value must be used throughout the life of a session. The Session ID must be globally and eternally unique, as it is meant to uniquely identify a user session without reference to any other information. This may be needed to correlate historical authentication information with accounting information. Example: aaa://host.example.com;protocol=diameter;-117302099;1
destinationHost	String	The fully qualified domain name of the HSS that the Sh subscriber profile Web service will send this request to. This input parameter is optional. Examples include: sipintel15.city.example.com or diameter.example.com This is an optional parameter. If specified, it must be the FQDN of destination Diameter node which handles the request.
destinationRealm	String	The realm that this subscriber belongs to. The destination realm is a required parameter and must be a fully qualified domain name. The value specified must match the specific *realmName* property that is defined in one of the `routex` properties in the `Diameter_Sh.properties` file. If the value does not match any of the specific routes, and a DEFAULT route entry is defined, the DEFAULT route will be used.
publicIdentity	String	Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22
subsReqType	int	Available subscription status values: 0 = SUBSCRIBE 1 = UNSUBSCRIBE
callbackUri	URI	The Web service URI used to deliver notification back to the subscriber; for example: http://example.com:7676/DHADiameterShNotifyService/services/DiameterShNotifyService
userId	String	If the IMS Application Server Notify Web service is protected, specify the user ID for that Web service.
password	String	If the IMS Application Server Notify Web service is protected, specify the password for that Web service.

Notify Web service methods

The notify Web service methods are used to send notifications to an external Web service.

For notifications, the IMS Application Server must implement a notification Web service. The Sh subscriber profile Web service will be a client to that notification Web service. The Sh subscriber profile Web service application will invoke the following methods based on the notification received from the HSS and the subscription records for Sh subscriber profile Web service clients:

- notifyRepositoryDataChange
- notifyIMSUserStateChange
- notifySCSCFNameChange
- notifyInitialFilterCriteriaChange

- notifyUserDataChange
- notifyPSIActivationChange

notifyRepositoryDataChange Web service method:

This Web service method will provide notification when the repository data changes.

Usage

This Web service method is used to receive notifications when the user profile RepositoryData for the associated subscriber changes in the HSS. The format of the RepositoryData is defined in the ShDataType.xsd schema. If the subscriber's profile has been deleted by the HSS, the ServiceData field in the RepositoryData object will be null.

Example

```
public void notifyRepositoryDataChange(RepositoryData newData,
                                       String publicIdentity)
                                       throws Exception {

  // application logic
}
```

Parameters

Parameter Name	Type	Description
newData	String	The modified or deleted RepositoryData received from the HSS notification request.
publicIdentity	String	This parameter is the publicIdentity associated with the updated RepositoryData. Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

notifyIMSUserStateChange Web service method:

This Web service method receives notification requests from the Sh subscriber profile Web service when the IMSUserState for the associated subscriber changes in the HSS.

Usage

This Web service method is used to receive notifications when the IMSUserState for the associated subscriber changes in the HSS. The possible values are shown in the following table:

Table 13. Return Values

Integer value	User state
0	NOT_REGISTERED

Table 13. Return Values (continued)

Integer value	User state
1	REGISTERED
2	REGISTERED_UNREG_SERVICES

Example

```
public void notifyIMSUserStateChange(int newState,
                                     String publicUserIdentity)
                                     throws Exception {

   // application logic
}
```

Parameters

Parameter Name	Type	Description
newState	int	One of the following integer values: 0: NOT REGISTERED 1: REGISTERED 2: REGISTERED_UNREG_SERVICES
publicUserIdentity	String	This parameter is the publicIdentity associated with the updated IMSUserState. A subscriber is allocated one or more public user identities, which are created by the home operator. A public user identity is either a SIP URL (as defined in RFC 3261) or a TEL URI (as defined in RFC 3966) and helps route a SIP request. Example: sip:Alice.Smith@example.com;transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

notifySCSCFNameChange Web service method:

This Web service method receives notification requests from the Sh subscriber profile Web service when the address of the Serving Call Session Control Function (S-CSCF) allocated to the subscriber changes in the HSS. Notifications can be received for both S-CSCF modifications and deletion.

Usage

This Web service method is used to receive notifications when the address of the Serving Call Session Control Function (S-CSCF) allocated to the subscriber changes in the HSS. The new URI of the S-CSCF and the associated publicIdentity are sent in the notification request. If the S-CSCF has been deleted by the HSS, the newURI parameter will be null.

Example

```
public void notifySCSCFNameChange(URI newURI,
                                  String publicIdentity)
                                  throws Exception {

    // application logic
}
```

Parameters

Parameter Name	Type	Description
newURI	URI	The modified URI of the updated Serving Call Session Control Function (S-CSCF) received from the HSS notification request. If the S-CSCF has been deleted by the HSS this parameter will be null.
publicIdentity	String	This parameter is the publicIdentity associated with the updated SCSCFName. Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

notifyInitialFilterCriteriaChange Web service method:

This Web service method receives notification requests from the Sh subscriber profile Web service when the filtering criteria for the associated subscriber changes in the HSS. Notifications can be received for both filter criteria modifications and deletions.

Usage

This Web service method is used to receive notifications when the InitialFilterCriteria for the associated subscriber and Application Server changes in the HSS. The format of the InitialFilterCriteria is defined in the ShDataType.xsd schema. If the InitialFilterCriteria for this subscriber has been deleted by the HSS, the value of the newCriteria parameter will be null.

Example

```
public void notifyInitialFilterCriteriaChange(InitialFilterCriteria newCriteria,
                                              String publicIdentity)
                                              throws Exception {

          // application logic
 }
```

Parameters

Parameter Name	Type	Description
newCriteria	InitialFilterCriteria	The modified or deleted InitialFilterCriteria received from the HSS notification request. If the InitialFilterCriteria for this subscriber has been deleted by the HSS, this value of this parameter will be null.
publicIdentity	String	This parameter is the publicIdentity associated with the updated InitialFilterCriteria. Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

notifyUserDataChange Web service method:

This Web service method receives notifications for the following types of user profile data as an XML string representation: Repository Data, IMSUserState, S_CSCFName, and InitialFilteCriteria. To receive notification data as an XML string, you must subscribe to the notifications using the Sh subscriber profile Web service subscribeUserData method. Notifications can be received for both data modifications and deletions.

Usage

This Web service method receives notification requests from the Sh subscriber profile Web service for the following types of user profile data: Repository Data, IMSUserState, S_CSCFName, and InitialFilteCriteria. The type of data received is indicated by the dataReference value; however, it is possible to receive updates for more than one data type in the XML file. The user profile data is returned as an XML string representation that adheres to the format defined in the ShDataType.xsd schema file. Data that has been deleted from the HSS is indicated by an empty element for that user data. For example, if the filter criteria for a particular subscriber has been deleted from the HSS, the InitialFilterCriteria element in the userData XML received will be empty or null.

Example

```
public void notifyUserDataChange(int dataReference,
                                 String userData,
                                 String publicIdentity)
                                 throws Exception {

        // application logic
 }
```

Parameters

Parameter Name	Type	Description
dataReference	int	Data reference values indicate the type of user data received from the HSS. 0 = Repository Data 11 = IMSUserState 12 = S_CSCFName 13 = InitialFilterCriteria 17 = MSISDN 18 = PSI Activation
userData	String	The XML string representation of the modified or deleted user profile data received in the HSS notification request. The XML data must comply with the `ShDataType.xsd` schema.
publicIdentity	String	This parameter is the publicIdentity associated with the updated UserData. Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

notifyPSIActivationChange Web service method:

This Web service method receives notification requests from the Sh subscriber profile Web service when the PSI Activation state for a specific service has changed in the HSS. Notifications will be received when the PSI Activation switches from Active to Inactive or vice versa.

Usage

This Web service method returns a value indicating the current state of activation (active = 1 or inactive = 0) of a service provided by an application. It also returns the Public Service Identity of that application. This function allows an application that wishes to use the services of another application the ability to know when that application is active and available to provide one or more services.

Example

```
public void notifyPSIActivationChange(int newActivation,
                                       String publicIdentity)
                                throws Exception {
    // application logic
   }
```

Parameters

Parameter Name	Type	Description
newActivation	int	Indicates the current state of application or service. Possible values are: 0 = INACTIVE 1 = ACTIVE
publicIdentity	String	This parameter is the publicIdentity associated with the updated RepositoryData. Specifies the public user identity or public service identity. This can be either a SIP URL or a TEL URL. Example: sip:Alice.Smith@example.com:transport=tcp sips:carol@ws1234.example.com sip:+358-555-1234567;postd=pp22@example.com;user=phone sip:alice;day=tuesday@example.com tel:+919-123-4567 tel:+358-555-1234567;postd=pp22

WSDLs

The Web Services Description Language (WSDL) is an XML-based language used to describe the interface of a Web service application. The WSDL is used to generate serializers and deserializers to read and write Web service SOAP requests and replies.

The WSDL separates the Web service interface definition (elements and namespace) from the Web service implementation (service and ports). The following WSDLs are available for use:

DiameterRfService.wsdl
Defines the Rf services used for offline charging. The Rf accounting Web service uses a simple request and reply structure to enable interaction between IMS Application Server applications and Charging Collection Function.

DiameterRoService.wsdl
Defines the Ro services used for online charging. It is structured in a fashion similar to the `DiameterRfService.wsdl` file. This interface allows

interactions between IMS Application Server applications and the Online Charging System (OCS) to exchange subscriber charging information. In addition, this interface also allows the application to subscribe for reauthentication notifications.

DiameterRoNotifyService.wsdl
Used by an IMS Application Server application to receive notifications from the Ro online charging Web service. Notifications are sent when the Charging Trigger Function (CTF) must reauthenticate with the OCS or when a reauthorization subscription expires. This WSDL file defines the interface that the customer must make available to receive these reauthorization notifications. You must implement the server side of this Web service to receive notifications. Diameter Enabler is the client side for this Web service. Notifications will only be sent if the IMS Application Server application has subscribed to receive them.

DiameterShService.wsdl
Defines the Sh services used for subscriber profile management. It is structured in a fashion similar to the `DiameterRfService.wsdl` file. This interface allows interactions between IMS Application Server applications and the HSS to exchange subscriber profile information.

DiameterShNotifyService.wsdl
Used by an IMS Application Server application to receive notifications from the Sh subscriber profile Web service when information regarding a specified subscriber has been changed. This WSDL file defines the interface that the customer must make available in order to receive these update notifications. You must implement the server side of this Web service to receive notifications. Diameter Enabler is the client side for this Web service. Notifications will only be posted if the application has made a subscription request for these notifications.

Result codes

This section provides information on the IBM WebSphere Diameter Enabler Component result codes.

The result code data field provides the following classes of errors:

Table 14. Diameter Result codes

Result codes and message descriptions
“1xxx (Informational)”
“2xxx (Success)” on page 208
“3xxx (Protocol errors)” on page 208
“4xxx (Transient failures)” on page 209
“5xxx (Permanent failure)” on page 209

1xxx (Informational)

Return codes within this category are used to inform the requester that a request could not be satisfied, and additional action is required on its part before access is granted.

Table 15. Informational result codes

Value	Message Name	Description
1001	DIAMETER_MULTI_ROUND_AUTH	This informational error is returned by a Diameter server to inform the access device that the authentication mechanism being used requires multiple round trips, and a subsequent request needs to be issued in order for access to be granted.

2xxx (Success)

Return codes within this category are used to inform a peer that a request has been successfully completed.

Table 16. Success result codes

Value	Message Name	Description
2001	DIAMETER_SUCCESS	The Request was successfully completed.
2002	DIAMETER_LIMITED_SUCCESS	When returned, the request was successfully completed, but additional processing is required by the application in order to provide service to the user.

3xxx (Protocol errors)

Errors within this category cause a Service Unavailable Exception to be sent to the calling application. The WebSphere Application Server log file will show an indication of these errors when they occur.

Table 17. Protocol error result codes

Value	Message Name	Description
3001	DIAMETER_COMMAND_UNSUPPORTED	The Request contained a Command-Code that the receiver did not recognize or support.
3002	DIAMETER_UNABLE_TO_DELIVER	This error is given when Diameter can not deliver the message to the destination.
3003	DIAMETER_REALM_NOT_SERVED	The intended realm of the request is not recognized.
3004	DIAMETER_TOO_BUSY	When returned, a Diameter node should attempt to send the message to an alternate peer.
3005	DIAMETER_LOOP_DETECTED	An agent detected a loop while trying to get the message to the intended recipient.
3006	DIAMETER_REDIRECT_INDICATION	A redirect agent has determined that the request could not be satisfied locally.
3007	DIAMETER_APPLICATION_UNSUPPORTED	A request was sent for an application that is not supported.
3008	DIAMETER_INVALID_HDR_BITS	A request was received whose bits in the Diameter header were set to an incorrect combination.

Table 17. Protocol error result codes (continued)

Value	Message Name	Description
3009	DIAMETER_INVALID_AVP_BITS	A request was received that included an AVP whose flag bits are set to an unrecognized value.
3010	DIAMETER_UNKNOWN_PEER	A CER was received from an unknown peer.

4xxx (Transient failures)

Errors within this category are used to inform a peer that the request could not be satisfied for the indicated reason. The WebSphere Application Server log file will show an indication of these errors when they occur.

Table 18. Transient failures result codes

Value	Message Name	Description
4001	DIAMETER_AUTHENTICATION_REJECTED	The authentication process for the user failed, most likely due to an incorrect password used by the user.
40002	DIAMETER_OUT_OF_SPACE	A Diameter node received the accounting request but was unable to commit it to stable storage due to a temporary lack of space.
4003	ELECTION_LOST	The peer has determined that it has lost the election process and has therefore disconnected the transport connection.
4010	DIAMETER_END_USER_SERVICE_DENIED	The OCS has denied this service request due to service restrictions. If the CCR contained Used-Service-Units, they will be deducted from the account if possible.
4011	DIAMETER_CREDIT_CONTROL_NOT_APPLICABLE	The OCS is indicating that the service can be granted free of charge. No CC session is required.
4012	DIAMETER_CREDIT_LIMIT_REACHED	The OCS is denying the request because the subscriber’s account does not contain sufficient credit. Used-Service-Units, if present in the CCR, will be deducted from the account, if possible.

5xxx (Permanent failure)

Errors within this category should not be retried because they will continue to fail until the implementation or configuration are changed. The WebSphere Application Server log file will show an indication of these errors when they occur.

Table 19. Permanent failure result codes

Value	Message Name	Description
5001	DIAMETER_AVP_UNSUPPORTED	The peer received a message that contained an AVP that is not recognized or supported and was marked with the Mandatory bit.
5002	DIAMETER_UNKNOWN_SESSION_ID	The request contained an unknown Session-Id.
5003	DIAMETER_AUTHORIZATION_REJECTED	A request was received for which the user could not be authorized.

Table 19. Permanent failure result codes (continued)

Value	Message Name	Description
5004	DIAMETER_INVALID_AVP_VALUE	The request contained an AVP with an incorrect value in its data portion.
5005	DIAMETER_MISSING_AVP	The request did not contain an AVP that is required by the Command Code definition.
5006	DIAMETER_RESOURCES_EXCEEDED	A request was received that cannot be authorized because the user has already expended allowed resources.
5007	DIAMETER_CONTRADICTING_AVPS	One or more Failed-AVP AVPs MUST be present, containing the AVPs that contradicted each other.
5008	DIAMETER_AVP_NOT_ALLOWED	A message was received with an AVP that should not be present. The Failed-AVP AVP must be included and contain a copy of the offending AVP
5009	DIAMETER_AVP_OCCURS_TOO_MANY_TIMES	A message was received that included an AVP that appeared more often than permitted in the message definition. The Failed-AVP AVP must be included and contain a copy of the first instance of the offending AVP that exceeded the maximum number of occurrences.
5010	DIAMETER_NO_COMMON_APPLICATION	This error is returned when a CER message is received, and there are no common applications supported between the peers.
5011	DIAMETER_UNSUPPORTED_VERSION	This error is returned when a request was received, whose version number is unsupported.
5012	DIAMETER_UNABLE_TO_COMPLY	This error is returned when a request is rejected for unspecified reasons.
5013	DIAMETER_INVALID_BIT_IN_HEADER	This error is returned when an unrecognized bit in the Diameter header is set to one (1).
5014	DIAMETER_INVALID_AVP_LENGTH	The request contained an AVP with an incorrect length. A Diameter message indicating this error MUST include the offending AVPs within a Failed-AVP AVP.
5015	DIAMETER_INVALID_MESSAGE_LENGTH	This error is returned when a request is received with an incorrect message length.
5016	DIAMETER_INVALID_AVP_BIT_COMBO	The request contained an AVP with which is not allowed to have the given value in the AVP Flags field. A Diameter message indicating this error MUST include the offending AVPs within a Failed-AVPAVP.
5017	DIAMETER_NO_COMMON_SECURITY	This error is returned when a CER message is received, and there are no common security mechanisms supported between the peers.
5030	DIAMETER_UNKNOWN_USER	The OCS does not recognize this user.
5031	DIAMETER_RATING_FAILED	The OCS was not able to correctly rate the service due to errors in one or more AVPs provided in the CCR.

Experimental result codes

Experimental result codes are vendor defined result codes that indicate success and error conditions. The Sh subscriber profile Web service and Home Subscriber Server (HSS) define several vendor-specific failure conditions.

Experimental result codes are included inside an Experimental-Result AVP, and the Result-Code AVP does not exist. The Sh application will retrieve the error message and return it through an exception. These exceptions are logged in theWebSphere Application Server `SystemOut.log` file.

Table 20. Sh application experimental result codes

Value	Message Name	Description
2001	DIAMETER_FIRST_REGISTRATION	The user is authorized to register this public identity. The user will be assigned an S-CSCF name.
2002	DIAMETER_SUBSEQUENT_REGISTRATION	The user is authorized to register this public identity. This user was assigned an S-CSCF name prior to this registration. A new S-CSCF name will not be selected.
2003	DIAMETER_UNREGISTERED_SERVICE	The public identity is not registered and has services that belong to an unregistered state. The user will be assigned an S-CSCF name.
2004	DIAMETER_SUCCESS_SERVER_NAME_NOT_STORED	The user has been unregistered. The S-CSCF name will be removed from the HSS.
2005	DIAMETER_SERVER_SELECTION	The user is authorized to register this public identity. An S-CSCF name has been assigned for services related to the previously unregistered state. The user may need to be assigned a new S-CSCF name.
5001	DIAMETER_ERROR_USER_UNKNOWN	The user that received this message does not exist.
5002	DIAMETER_ERROR_IDENTITIES_DONT_MATCH	The HSS cannot identify the user's corresponding private identity for the public identity submitted in the message.
5003	DIAMETER_ERROR_IDENTITY_NOT_REGISTERED	A new public identity submitted a query for LocationInformation. The user will not receive LocationInformation because its public identity has not been registered.
5004	DIAMETER_ERROR_ROAMING_NOT_ALLOWED	The user is not allowed to roam in the visited network.

Table 20. Sh application experimental result codes (continued)

Value	Message Name	Description
5005	DIAMETER_ERROR_IDENTITY_ALREADY_REGISTERED	This identity has been assigned to a server prior to this request. The registration status will not overwrite the existing identity.
5006	DIAMETER_ERROR_AUTH_SCHEME_NOT_SUPPORTED	The authentication scheme is not supported.
5007	DIAMETER_ERROR_IN_ASSIGNMENT_TYPE	This identity has been assigned to the same server. The registration status does not allow this server assignment type.
5008	DIAMETER_ERROR_TOO_MUCH_DATA	New data will be discarded because the volume of data being pushed to the receiving entity exceeds the capacity.
5009	DIAMETER_ERROR_NOT_SUPPORTED_USER_DATA	The S-CSCF has informed the HSS that the received subscription data contained unrecognizable or unsupported information.
5010	DIAMETER_MISSING_USER_ID	The message cannot be processed because the HSS has informed the S-CSCF that the message did not contain a private identity or a public identity.
5011	DIAMETER_ERROR_FEATURE_UNSUPPORTED	The received request indicates the origin host requests that the command pair would be handled using an unsupported feature by the destination host.
5100	DIAMETER_ERROR_USER_DATA_NOT_RECOGNIZED	The data required in the XML schema does not match the data specified in the HSS.
5101	DIAMETER_ERROR_OPERATION_NOT_ALLOWED	The HSS has ignored the operation because the Sh-Update request does not includes service data.
5102	DIAMETER_ERROR_USER_DATA_CANNOT_BE_READ	The requested user data cannot be read.
5103	DIAMETER_ERROR_USER_DATA_CANNOT_BE_MODIFIED	The requested user data cannot be updated.
5104	DIAMETER_ERROR_USER_DATA_CANNOT_BE_NOTIFIED	The requested user data cannot be notified on changes.

Table 20. Sh application experimental result codes (continued)

Value	Message Name	Description
5105	DIAMETER_ERROR_TRANSPARENT_DATA_OUT_OF_SYNC	The request to update the repository data at the HSS cannot be completed because the requested update is based on an out-of-date version of the repository data. Two actions can cause this error: • The sequence number in the Sh-Update request message does not match the immediate successor of the associated sequence number stored for that repository data in the HSS. • It is also used when the IMS Application Server creates a new set of repository data when the repository data exists in the HSS.
4100	DIAMETER_USER_DATA_NOT_AVAILABLE	The requested user data is not available at this time to satisfy the requested operation.
4101	DIAMETER_PRIOR_UPDATE_IN_PROGRESS	The request to update the repository data at the HSS could not be completed because the related repository data is currently being updated by another entity.

Chapter 11. 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 21. Change history for the product documentation

Edition	Date	Changes
First Edition	October 2007	First issue of the product documentation.
Second Edition	March 2008	Fix pack updates.
Third Edition	April 2009	Minor updates to reflect compatibility between IBM WebSphere software for Telecom, version 7.0 products, and WebSphere IMS Connector, version 6.2.

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:

AIX
DB2
IBM
pSeries
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 IP Multimedia Subsystem Connector
IBM WebSphere IP Multimedia Subsystem Connector
Version 6.2

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