IBM WebSphere Transformation Extender

Transformation Extender Introduction

Version 8.1

Note
Before using this information, be sure to read the general information in "Notices" on page 41.

October 2006

This edition of this document applies to IBM WebSphere Transformation Extender Version 8.1; and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email DTX_doc_feedback@us.ibm.com. We look forward to hearing from you.

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. WebSphere Transformation Extender introduction

WebSphere Transformation Extender is a powerful, transaction-oriented, data integration solution that automates the transformation of high-volume, complex transactions without the need for hand-coding. This provides enterprises with a quick return on investment. This product supports EDI, XML, SWIFT, HIPAA and other standards-based B2B integration, as well as the real-time integration of data from multiple applications, databases, messaging middleware and communications technologies across the enterprise.

The business role of WebSphere Transformation Extender

WebSphere Transformation Extender is integrated with other components of the IBM WebSphere Enterprise Integration Suite for complete operational and transactional data integration across the enterprise. It provides the following:

- highly automated transformation and routing of complex data across many points of integration in real time to support high-message volumes
- enterprise-wide interoperability supported by a Services-Oriented Architecture (SOA) for seamless connectivity and interoperability across back-office systems
- support for high-performance, event-driven, transactional environments to ensure completion and validation of transactions in real time
- seamless integration across the development, data, and production layers of the enterprise, leveraging existing IT infrastructures
- out-of-the-box solutions for industry standards and regulatory compliance for operational and transactional data integration

Chapter 2. Accessibility features for WebSphere Transformation Extender

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully. IBM strives to provide products with usable access for everyone, regardless of age or ability.

Accessibility features

The following list includes the major accessibility features in WebSphere Transformation Extender. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.

Tip: The WebSphere Transformation Extender Eclipse-based help system is accessibility-enabled for the IBM® Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard navigation

The following product components use standard Microsoft® Windows® navigation keys:

- Design Studio applications
- Command Server
- Launcher Monitor
- Snapshot Viewer

Resource Registry, Management Console, and Launcher Administration use standard Java navigation keys.

Vendor software

WebSphere Transformation Extender includes certain vendor software that is not covered under the IBM license agreement. IBM makes no representation about the accessibility features of these products. Contact the vendor for the accessibility information about its products.

Related accessibility information

The following applications have some additional keyboard functionality. The key combinations are documented in the application-specific help topics or PDFs.

- Database Interface Designer
- Integration Flow Designer
- Launcher Management Tools
- Map Designer
- Type Designer

IBM and accessibility

See the *IBM Accessibility Center* for more information about the commitment that IBM has to accessibility.

Chapter 3. Product overview

WebSphere Transformation Extender performs transformation and routing of data from source systems to target systems in batch and real-time environments. The sources may include files, relational databases, MOMs (message-oriented middleware), packaged applications, or other external sources. After retrieving the data from its sources, the WebSphere Transformation Extender product transforms it and routes it to any number of targets where it is needed, providing the appropriate content and format for each target system.

The WebSphere Transformation Extender product delivers the following:

- connectivity to a wide range of mainframe, legacy, and enterprise applications, databases, messaging systems, and external information sources
- a comprehensive library of more than 120 pre-built functions to reduce development time and simplify specification of rules for validation, transformation, and routing
- multiple execution options to support right-time, right-style transformation-whether it is batch, real-time, or embedded
- enterprise-class capabilities for development, deployment, and maintenance plus high-availability platform support.

This reduces on-going administration and implementation risks and delivers results sooner than hand-coding.

WebSphere Transformation Extender is a run-time environment that implements both event-driven and command-driven integration for:

- Enterprise Resource Planning (ERP) applications
- Customer Relationship Management (CRM) applications
- Supply Chain Management (SCM) applications
- other off-the-shelf applications
- customer-developed, in-house applications
- direct, business-to-business integration
- business community integration
- integration with trade networks, net markets, and business service providers

In WebSphere Transformation Extender terms, an integration solution consists of a system of related interfaces, implemented as rule-based maps, that consume and produce data in support of a business process. Maps define interfaces between applications, data stores, middleware, and other maps.

Chapter 4. Developing application interfaces

The WebSphere Transformation Extender Design Studio development environment includes a robust set of integrated tools designed to solve varied application integration problems. Except for scenarios in which the WebSphere Transformation Extender SDK is used, you can use these tools to solve integration problems non-invasively, without modifying your applications.

WebSphere Transformation Extender products are designed to fit your interface architecture-whether it is hub-based or distributed. Some companies prefer hub-models with a centrally managed integration server. Other organizations choose to distribute integration servers to reside on the same platform as the applications they service. WebSphere Transformation Extender products can be used in either configuration.

To design an application interface, you need to understand the content of each data source and target, as well as the middleware, transports, and data services needed to communicate with corresponding sources and targets. You also need to define rules for transforming data content from inputs to outputs.

As you develop more and more interfaces, you will want to reuse metadata and integration rules. Additionally, you will want to formulate business process models: interface systems consisting of integration components that are configured to work together. These systems also need to be reusable.

The WebSphere Transformation Extender Design Studio is used to develop new integration components and to reuse existing components so that you can leverage your development efforts across integration projects.

Modeling the data of business objects

Although you can use the WebSphere Transformation Extender Design Studio to represent the properties of individual fields, elements, and records, its strength lies in defining complete *business objects*. In the WebSphere Transformation Extender Design Studio, business objects are the substance of business transactions such as the following: health claims, invoices, bank transfers, airline reservations, ship notices, or telephone bills.

Business objects that include interface syntax, semantics, and structure are implemented in the WebSphere Transformation Extender products as *type trees*, which may be pre-defined (in the case of standards supported out-of-the-box by WebSphere Transformation Extender), imported from metadata, or graphically defined using the WebSphere Transformation Extender Type Designer. Once defined, they can be re-used to describe interface content for multiple sources and targets.

- Use the Type Designer to define properties for text or binary data, different character sets, data structures, and semantic validation rules. The resulting type definition is enforced automatically and transparently, when the WebSphere Transformation Extender system executes.
- When external metadata exists, you can reduce the effort of defining business objects by using application importers. For example, you can use the COBOL Copybook Importer to generate type trees from COBOL record structures, or the

Database Interface Designer to generate type trees from database catalog entries. You can use the DTD Importer to generate type trees from XML Document Type Definitions (DTDs). PeopleSoft, SAP R/3, and Siebel users can generate type trees from application-maintained metadata, using WebSphere Transformation Extender-supplied importers.

- WebSphere Transformation Extender products provide a repository of predefined type trees for popular e-commerce, healthcare, insurance, and financial standards, including ASC X12, EDIFACT, HIPAA, HL7, SWIFT, ISO 15022, and others.
- Using the Type Tree Maker, which is also part of the WebSphere Transformation Extender Design Studio, you can build your own importer to automate the capture of metadata definitions from machine-readable sources.

Regardless of method used, the result is a set of reusable business object definitions encapsulated in a graphical, tree-like structure called a *type tree*. For more information about type trees, refer to ″Using Type Trees″.

Defining transformation rules

Complete definitions of data objects and the rules for their transformation are embodied in an WebSphere Transformation Extender *map*. Map objects are created using the Map Designer, a graphical tool that makes complex transformations easier to define and maintain. Maps are analyzed, compiled, and tested directly in the WebSphere Transformation Extender Design Studio environment.

A map object can implement a wide range of interface functions, from simple transformation to sophisticated integration solutions involving multiple, heterogeneous inputs and outputs, rule-based routing, and complex interface structures.

The WebSphere Transformation Extender Design Studio transformation server processes interface data without pre-formatting or preprocessing. You can compare, move, and transform data using any combination of sources and targets, without regard for underlying interface formats. The graphical environment of the Design Studio also delivers complete integration solutions that require no programming to complete the job. Complex interface problems that involve multiple input sources and multiple output targets can be handled in a single map. There is no need to break up a unit of work into multiple steps unless the business problem requires it.

The Design Studio's comprehensive set of integration rules enables you to go beyond transformation to construct and route entirely new business objects. You can apply rules to route business objects to appropriate targets based on data content, runtime conditions, or any other criteria you choose. You can apply rules to filter, scrub, calculate, validate, parse, substitute, expand, change character set or otherwise convert data from multiple inputs to multiple outputs. All of these capabilities supplement simple, drag-and-drop mapping where needed.

With the Design Studio, data sent to an output does not have to come from an input. You can compute data, look up something in a database, sort, extract, merge, or use any of dozens of WebSphere Transformation Extender functions to create output, where appropriate.

You can also map outputs to each other. For example, you can transform data that was generated at the top of an output to the bottom of the same output so that you can summarize, tally, or validate what has been created. You can also map an

output that was generated in one output to another output. For example, you can generate a message as one output and archive all or part of it as another output.

Multiple input and output types can be used within a single map for any number of sources (adapters) in an any-to-any fashion. WebSphere Transformation Extender is also designed for one-to-many, many-to-one and many-to-many processing. This flexibility provides unconstrained support for common integration needs, such as creating a sales distribution document and an invoice from the same purchase order and updating the inventory database at the same time.

Defining data sources and targets

A data *source* is defined in an WebSphere Transformation Extender system by specifying a resource adapter for an input. A data *target* is defined by specifying a resource adapter for an output. WebSphere Transformation Extender includes resource adapters for files, memory, databases, messaging middleware products, HTTPS, FTP, e-mail, and more. Application adapters are also available for interfacing with popular commercial applications. Furthermore, you can develop your own adapters to integrate directly to new types of sources and targets by using the WebSphere Transformation Extender open adapter interface.

The way in which business objects are defined and the way in which map rules are specified is generally independent of the actual data source or data target due to the abstraction afforded by the type tree implementation. Business objects deal with the *format* and *content* of the data, serving to answer questions such as ″What should the data look like?″ and ″What should the value of the data be?″ Maps deal with the transformation logic required to create new business objects from existing ones. By contrast, resource adapters answer such questions as ″Where should the data come from or go to?″ and ″How does the application or resource expect the map to send or receive the data?″

During interface design, you specify a resource adapter for each input and for each output. This determines the default adapters to be used during execution. You can always change the source or target by using the process modeling component, the Integration Flow Designer, or at execution time. WebSphere Transformation Extender transformation solutions are resource-independent.

You can also use resource adapters in map rules. For example, you can use database adapters in a map rule to look up cross-reference data stored in a database. When the resource adapter you want to use varies, depending upon the data content, you can dynamically allocate the resource from within a map rule. The transformation solutions from WebSphere Transformation Extender software can be reconfigured.

Defining process models

The Integration Flow Designer is the graphical process modeling component of the WebSphere Transformation Extender Design Studio. Use the Integration Flow Designer during development to model a set of logically related maps called a *system.* Systems can be used as a focal point for interfacing to other development components and for preparing maps for execution.

Formulating business process models

Using the Integration Flow Designer, you formulate a business process model as a system. System components can be maps or other systems. A transformation

component and its inputs and outputs are represented in graphical form. Data flow links are automatically diagrammed from information about the resource adapters assigned to inputs and outputs. During development, you can navigate to an underlying map where integration rules are defined and to the underlying definitions of input and output business objects.

Defining resource connectivity

When you use the Integration Flow Designer to specify resource adapters for map components, links are automatically established when the target of one component is used as the source of another. You can see the flow of information among system components as you design.

Defining event rules

The Integration Flow Designer can be used to configure the event rules that determine the conditions under which a map will run.

A map component can be initiated based on one or more time, file, database application, or message events. For example, you can specify the insertion of data into a database table to be a map trigger. Alternatively, you can also specify the synchronous coordination of multiple events, such as ″Trigger this map every day at 4:00 P.M. if there is a message on my queue″. Event management can also include wildcard matching for events across inputs and outputs. For transaction-based transformations, there are also rollback, delete, and retry options.

Configuring map settings

The Integration Flow Designer includes many options for configuring each map such as: priority, event expiration settings, audit logging, paging, multi-threading options, and more. This allows you to configure your production systems for optimal performance.

Managing systems

To prepare a system for deployment, you will want to build, analyze, and possibly port, all the map components in that system. Use the Integration Flow Designer to do this.

Consistent control information can be generated from the graphical diagram of system components. For example, you can generate map command files for a Command Server or control files for an Launcher. This information can be used to distribute transformation components and subsystems across different servers to execute systems as an integrated whole.

Chapter 5. Deploying what you develop

WebSphere Transformation Extender products bring industrial-strength features to a wide range of applications, from event-driven process automation to large-scale batch transformations.

Executing in test and production environments

During the development process, test execution takes place in the same WebSphere Transformation Extender Design Studio environment in which you develop your integration solution. Testing can be performed offline, using the Design Studio, even when the planned execution platform, deployment mode, or adapter assignments differ from those used in the test environment.

You can configure your environment with multiple servers at multiple sites, and on multiple platforms. Alternatively, you can select a single server as your deployment environment. For example, you can develop a system that includes all the information about how to process payments. You can then provide this map to multiple locations in your company, for execution on different server platforms. The distributed sites do not need to know anything about your development environment.

Executing on multiple platforms

WebSphere Transformation Extender maps are portable to a wide selection of different execution platforms. A map is defined once, using the WebSphere Transformation Extender Design Studio, and that same map can execute on any of the popular platforms that the WebSphere Transformation Extender products support.

Chapter 6. WebSphere Transformation Extender products

Product architecture

The WebSphere Transformation Extender product is client/server software composed of a client-based design environment, several server-based execution options, and administration and operation capabilities that can be run on the client or server.

Runtime services

WebSphere Transformation Extender includes several subsystems that provide essential runtime services to integration solutions:

- Launcher

  The WebSphere Transformation Extender Launcher monitors and evaluates runtime events and triggers map execution according to associated rules.
- Command Server

  The WebSphere Transformation Extender Command Server supports command-driven and scripted invocation of integration solutions.
- Engine

  The WebSphere Transformation Extender engine executes maps in the Launcher and Command Server environments.
- Launcher Agent

  The WebSphere Transformation Extender Launcher Agent implements a secure listener function outside of the corporate firewall that detects and routes inbound HTTP requests to Launcher Servers inside the firewall. The Secure Adapter Collection is an option that provides support of the transport layer and secure sockets layer (SSL) security.
- WebSphere Transformation Extender resource adapters

  The WebSphere Transformation Extender resource adapters bind logical map interfaces with applications, databases, middleware services, and other resources, hiding the details of APIs, marshalling, and low-level interaction protocols.

Collectively, these services simplify the creation of seamless, end-to-end, integration solutions from the firewall to the back office, by implementing a consistent architecture and supporting a single integration methodology for any combination of applications, interface formats, interprocess communication mechanisms, and integration flow logic.

Software Development Kit (SDK)

The WebSphere Transformation Extender Software Development Kit (SSDK) and related options provide the means to include transformational capabilities within a custom application or environment and to extend WebSphere Transformation Extender product capabilities using an adapter toolkit. When used with an application server or web server, the components in the SDK can be used to facilitate web-based integration scenarios.

The following components comprise the Software Development Kit:

- Software Development Kit Application Programming Interfaces (APIs)

- Software Development Kit Options
- Adapter Toolkit

Software Development Kit Application Programming Interfaces (APIs)

At the center of the SSDK are object-oriented application programming interfaces (APIs) that are available for C, Java, RMI, COM, and CORBA environments. These APIs provide common functionality that is represented in language-specific methods and objects. All APIs within the WebSphere Transformation Extender SSDK contain the same underlying objects and methods. The APIs support multithreading to ensure high performance by allowing multiple instances of maps that can be run in parallel.

Map and card settings and parameters that can be changed in the Map Designer or Integration Flow Designer can also be passed to the map at runtime through these APIs, thus allowing runtime control of specific instances of the map. For example, an application variable can be used to control audit settings for a particular map or to pass in a filename to be accessed.

Software Development Kit options

Development Kit options, such as the Control for BEA WebLogic and the Plug-in for IBM WebSphere MQ Integrator, integrate WebSphere Transformation Extender maps with other application integration and business process management products.

Adapter toolkit

The Software Development Kit includes a toolkit with an object-oriented interface for developing robust custom adapters that can:

- tightly integrate with WebSphere Transformation Extender products using the XML registration file
- participate in connection pooling
- provide an event listener

Design environment - The Design Studio

The Design Studio provides the means for developing event-driven application-to-application (A2A) integration, business-to-business (B2B) integration, and consumer-to-business (C2B) application integration. The Design Studio is the design component of WebSphere Transformation Extender used to model and test integration solutions. The following documentation describes Design Studio components and functionality:

- ″Client Components″
- ″Server Components″
- ″Management and Monitoring Tools″

Client components

The WebSphere Transformation Extender data integration solutions are created using the Design Studio development environment. These client components operate in a Windows environment and provide the graphical interface for managing the integration of applications and business processes.

The Design Studio is the client interface into an extensive array of functionality that is available to create application integration solutions. Application integration solutions created with the Design Studio are executed on transformation servers specific to the WebSphere Transformation Extender product being used. Separation of design and execution provides exceptional user flexibility in implementing enterprise-wide integration solutions. One Design Studio can deploy integration solutions to multiple servers or multiple Design Studios can be used to create integration solutions for a single production environment.

The Design Studio consists of multiple, user-friendly, GUI-based applications that are Windows-based. The client components in the Design Studio can be used to:

- Define source and target data structures (using the Type Designer)
- Import database object metadata and identify characteristics of those objects to meet mapping and execution requirements (using the Database Interface Designer)
- Define transformation rules (using the Map Designer)
- Design systems of maps to model data integration processes (using the Integration Flow Designer)
- Develop and test maps and systems (using the Map Designer and the Integration Flow Designer)
- Deploy maps and systems to the execution environment (using the Integration Flow Designer)

The following is a list of client components in the Design Studio:

- Type Designer
- Map Designer
- Database Interface Designer
- Integration Flow Designer
- Type Tree Maker
- Command Server
- Resource Registry

Type Designer

The WebSphere Transformation Extender Type Designer is the data object modeling component used to perform the following functionality and more:

- Create and manage type trees that define properties for data structures.
- Define containment of data.
- Create data validation rules.

Using type trees

A *type tree* is a graphical data dictionary that contains metadata definitions of the structure of the inputs and outputs that are the source and target data for integration solutions. It provides visual cues for understanding the structure of an object and supports point-and-click techniques enabling you to ″drill down″ to uncover an object's complete definition.

Each data object in the input or output data is defined as a reusable object in a type tree that uses a sophisticated set of properties to describe attributes such as length, justification, possible values, delimiters, and the order of the data objects that comprise a complex data object. The Type Designer is used to define

properties for text or binary data, different character sets, data structures, and semantic validation rules. The sophisticated type properties and type model of WebSphere Transformation Extender enable the precise definition of highly complex data, such as EDI, SWIFT, XML and legacy data structures.

Although the discussion of type trees appears in this Type Designer section, the type trees you use can originate from any of the following possible sources:

- They can be manually created using the Type Designer.
- They can be automatically created by a type tree importer or the Database Interface Designer-each of which generates trees based upon known metadata, such as XML DTD/Schema, database catalogs, SAP R/3 IDocs, or COBOL Copybooks.
- They can be predefined type trees that are included as part of IBM WebSphere TX Packs that support particular standards-such as ASC/X12, UN/EDIFACT, SWIFT, or HIPAA.
- They can be created by the Type Tree Maker-a tool that enables creation of custom importers to automate the capture of metadata definitions from other machine-readable sources.

For more information about these sources, refer to the Type Designer, Database Interface Designer, the specific WebSphere DataStage TX Pack, and Type Tree Maker documentation, respectively.

Using type tree importers

The Type Designer includes importers to automatically generate type trees (data object definitions) from known metadata, such as eXtensible Markup Language (XML) Document Type Definitions (DTDs) and Schema, COBOL Copybooks, and Enterprise Java Beans (EJBs). Each importer provides a wizard-driven interface to walk you through the process of reading the metadata and create the type tree.

Some importers are available in the Type Designer. Additional importers, for data formats such as SAP R/3 IDocs and PeopleSoft Component Interface, are provided as part of the WebSphere Transformation Extender Packs for WebSphere Transformation Extender.

For more information about the type tree importers, refer to the Type Tree Importers documentation.

Comparing type trees for differences

You can use the type tree differences feature in the Type Designer to compare two type tree files. When the analysis is complete, both trees appear. Source compare differences are distinguished by either blue or red text. When a type is present in one tree but not in the other, the text is blue. When a type is present in both trees, but they each have different properties, components, or restrictions, the text is red. There is a ″step-through″ facility that cycles through each of the differences in turn. In extremely large, complex type trees, this can save a great deal of time.

For more information about the type tree differences functionality, refer to the Type Designer documentation.

Map Designer

The WebSphere Transformation Extender Map Designer is the modeling component used to formulate transformation and business rules. The Map

Designer uses definitions of data objects created in the Type Designer as inputs and outputs. The Map Designer provides functionality for specifying rules for transforming and routing data, as well as the environment for analyzing, compiling, and testing the maps that are developed.

This process is facilitated by convenient ″from″ and ″to″ windows, drag-and-drop techniques, and spreadsheet-like rules. Mapping rules are added using logical statements. The Map Designer provides a rich set of more than 100 predefined functions for operations such as conditional testing, table lookups, mathematical functions, character string parsing, and data extraction.

Using maps

A *map* is the embodiment of complete definitions of data objects and the rules for their transformation. A map can implement a wide range of integration functions-from simple transformation to sophisticated integration solutions involving multiple, heterogeneous inputs and outputs, rule-based routing, and complex interface structures.

Maps are created using the Map Designer, a graphical tool that makes complex transformations easier to define and maintain. Maps are analyzed, compiled, and tested directly in the Design Studio environment.

Any number of inputs or outputs can be used within a single map in an any-to-any fashion. A map contains a *card* for each input and output of the transformation. The following example is a map containing one input card and one output card, showing the transformation of a LineItem element and its corresponding PO element from an XML document to a PO1 loop in an X12 purchase order transaction set (850).

The capability for mapping from multiple inputs to multiple outputs gives WebSphere Transformation Extender products unique transformation power. Each input to and output from a map is associated with a specific data object definition and a resource adapter. Data object definitions are reusable—the same definition can apply to one or more inputs; to both input and output for the same map; or to different maps. When a data object is used as input, data is validated to ensure the data conforms to the data object definition. When used as output, the data object is constructed automatically, in its entirety.

Input objects are not consumed when used in a transformation rule. This allows for a one-to-many relationship between an input and its outputs. An output can be constructed from any number of zero or more inputs. This creates a one-to-many relationship between an output and its inputs. Taken together, WebSphere Transformation Extender products provide many-to-many transformation support between inputs and outputs.

Data object definitions, data maps, and resource adapters are all separately managed objects. Each object is reusable in any number of integration scenarios. The result is flexibility, ease of use, and reduced maintenance as integration requirements change.

Defining server platforms for maps

The map definition is completely portable between all server platforms. Simply select the target platform from a list when building the map to generate a compiled map file that is optimized for the specified platform.

Exporting maps

As with the Type Designer, the Map Designer is able to export its rules and execution settings to a flat file in XML format.

Comparing maps for differences

As with the Type Designer, the Map Designer also has a graphical source compare feature that allows you to compare two map source files.

For more information about maps, cards, and the Map Designer, refer to the Map Designer documentation.

Database Interface Designer

The WebSphere Transformation Extender Database Interface Designer is the modeling component used to import metadata about queries, tables, and stored procedures for data stored in relational databases. The Database Interface Designer identifies characteristics, such as update keys and database triggers, of those objects to meet mapping and execution requirements.

The Database Interface Designer is used to:

- Specify the databases to use for a source or target, along with the user IDs and passwords required for the connection.
- Define query statements.
- Automatically generate type trees for queries, stored procedures, input and output parameters, or tables.
- Define tables for which any inserts, updates, or deletions would trigger maps using the Launcher.

If connectivity to the database server is not available from the development platform, the mtsmaker utility is provided to execute on the server platform. This utility creates a Type Tree Maker script file that can be transferred to the development environment to generate the type trees.

For more information about the Database Interface Designer, refer to the Database Interface Designer documentation.

Integration Flow Designer

The WebSphere Transformation Extender Integration Flow Designer is the modeling component used to define and manage data integration processes. The Integration Flow Designer is used to define interactions among maps and systems of maps, to validate the logical consistency of workflows, and to prepare systems of maps to run.

Using systems

A system is a set of maps (or subsystems) that are combined to form a data integration process flow. Systems are defined in the Integration Flow Designer.

Systems are defined as having one of two execution modes-Launcher or Command Server, depending upon how they will be deployed. If a system is developed for the Launcher, the system will include definitions of events based upon the map execution to be triggered. As mentioned previously, these events may be time events (for example, execute every 10 minutes or every Monday at 2:43 AM),

source events (for example, a message appearing on a queue, a row being updated in a database), or compound events comprised of some combination of time and/or source events.

The set of events that initiates a map within a system is called a *watch*. Each event that contributes to the initiation of a map is called a *trigger*. When the resource (file, database, or message queue) that is an input to a map changes state (that is, it is created or modified), this input event can cause the Launcher to initiate a map. Adapters that can be used to trigger maps have a listener function to detect the appropriate external resource change.

The Launcher starts maps based upon these triggers. There may be multiple triggers that must collectively exist before a map is initiated. The Launcher manages the coordination of these events to ensure that the correct set of circumstances has occurred before a map is initiated.

For more information about the Integration Flow Designer, refer to the Integration Flow Designer documentation. For more information about the Launcher, refer to the Launcher documentation.

Using links

A link is a connection that the Integration Flow Designer creates between two components in a system. Links represent the direction of data flow between the system components at execution time, thus showing the source and target dependencies between system components.

Managing systems

The Integration Flow Designer provides clarity and control to manage the more complex systems built using WebSphere Transformation Extender products. It also reduces the administrative burden and simplifies the non-development activities associated with interface development. Some of the key features of the Integration Flow Designer are:

- Graphical data flow design

  Visual data flows are created to aid in the design of integration solutions. You can use a graphical palette to produce system definition diagrams and to easily navigate among the WebSphere Transformation Extender Design Studio tools. Point-and-click techniques can be used to select the map and subsystem components to place in a specific integration flow diagram.

  Using the Integration Flow Designer, you can also define specific map settings that take effect at run time. Select the maps and the data flow between them is automatically determined from the inputs and outputs. The **Override** tool in the Integration Flow Designer toolbox can be used to modify the inputs and outputs to achieve the desired data flows. These modifications do not affect the source map. This allows the same map to be re-used in multiple data flows.
- Analysis of integration provided for logical consistency

  The Integration Flow Designer includes a System Analyzer that checks your system definitions for logical consistency to ensure they can be executed. The System Analyzer detects any inconsistencies you may have introduced in the definition while experimenting with system models.
- Capability to build and port entire systems with simple mouse-clicks

After a system has been designed, the Integration Flow Designer offers another benefit that saves time when preparing a system for operation. One mouse-click builds and deploys all of the integration objects and supporting data (such as lookup files and scripts).

Parameters for different servers representing different operating environments can be defined within the Integration Flow Designer.

After the appropriate server definitions have been created, it is then possible to produce, using the wizard, automated deployment definitions that are specific to each server.

It is possible to specify, within a script, that either all maps are built and transferred or that only a specific subset of the maps in a system are build and transferred. You can also include additional files required to support the system's operation in the transfer.

For more information about the Integration Flow Designer, refer to the Integration Flow Designer documentation.

Type Tree Maker

The WebSphere Transformation Extender Type Tree Maker is a scripting tool that automates the capture of metadata from machine-processable sources to create graphical metadata in the form of type trees.

For more information about the Type Tree Maker, refer to the Type Tree Maker documentation.

Command Server

The WebSphere Transformation Extender Command Server is the Windows-based version of the server component that provides for command-driven execution. It is included as part of the Design Studio for testing maps within the Map Designer development environment.

For more information about the Command Server, refer to the *Command Server documentation*.

Resource Registry

The WebSphere Transformation Extender Resource Registry is an application providing a resource alias repository that is used to abstract parameter settings using aliases that resolve at execution time to specific resources within the enterprise. These variables are automatically resolved to the actual resources, such as directory paths, database names, message queues, and server names, at execution time, making it possible to use the same map and system definitions in different deployment environments.

For more information about the Resource Registry, refer to the Resource Registry documentation.

Server components

Several mechanisms, including a Java API that enables Java-based interfaces to maps, are available for integration components. All of the methods share a common core engine and adapters. The main methods of execution are:

- Command-driven

  This provides command-driven execution and is used primarily for batch processing. The WebSphere Transformation Extender Command Server is used for this method.

- Event-driven

  This provides event-driven execution, in which systems of maps can be coordinated to provide a scalable, distributed, multithreaded, execution environment. The WebSphere Transformation Extender Launcher is used for this method.
- Application-embedded model

  This model allows maps to be embedded in third-party applications. The WebSphere Transformation Extender Development Kit is used for this method.

Management and monitoring tools

The management and monitoring tools are a set of graphical applications that provide a way to manage and view processes running in the Launcher. These graphical tools are available on Windows- and UNIX-based platforms and can be used with IBM WebSphere DataStage TX Launchers that are either local or remote. The following tools are described in this documentation:

- Launcher Administration
- "Simple Network Management Protocol (SNMP) support"
- Management Console
- Launcher Monitor
- Snapshot Viewer

Launcher Administration

The WebSphere Transformation Extender Launcher Administration is the administrative interface to the IBM WebSphere DataStage TX Launcher, from which the user specifies deployment directories, configures users and user access rights, specifies listening ports, and defines properties for Java Remote Method Invocation (RMI).

Configurations can be defined for:

- automatic system startup when the Launcher service is started
- connection ports for the Management Console
- resource resolution directives as defined in the Resource Registry

Simple Network Management Protocol (SNMP) support

Using the WebSphere Transformation Extender SNMP Agent, you can monitor and manage the status of the Launcher and associated map events. The SNMP Agent can be used to configure thresholds, monitor Launcher performance, and assist in identifying failure situations.

SNMP support is provided with the Launcher on all Windows and UNIX platforms. It supports the following traps:

Trap Warnings and Errors Generated

Map failure
It signals when a map fails.

State and listener change
It signals a change in the state of the IBM WebSphere DataStage TX Launcher or listeners.

Launcher/resource connection failure
It notifies the SNMP management software that the connection to the Launcher and the resource could not be established.

Thresholds
It signals when a map or connection pending state threshold has been met.

Additionally, an SNMP adapter allows for custom-defined traps to be issued from within the business logic of a map.

For more information about the SNMP Agent, refer to the *SNMP Agent documentation*.

Management Console

The WebSphere Transformation Extender Management Console is the management and monitoring interface for the Launcher, from which the user can start, stop, pause, and resume the system, as well as view information about the status of the Launcher and maps that are currently running. The Management Console also permits centralized, remote management of all Launcher installations.

Separate connections are defined for each Launcher installation. Each Launcher then appears in the main window of the Management Console, and each can be managed from the single, remote interface.

From the Management Console, it is possible to remotely stop, start, and pause systems. Launcher debugging can also be enabled to assist in operational troubleshooting of Launcher operations.

The Management Console provides a variety of statistics about the Launcher, including the length of time it has been running and the number of maps that have been processed.

- Summary: provides a general view of performance and status information.
- Status: contains active and pending system status information.
- History: contains a rollup of the total number of maps that have been run, the number of successes and failures, and so on.
- Configuration: contains the configuration parameters for the Launcher.

For more information about the Management Console, refer to the Launcher documentation.

Launcher Monitor

The WebSphere Transformation Extender IBM WebSphere DataStage TX Launcher Monitor is a GUI that provides a dynamic, detailed view of watches (single map instances) as they run. It also provides the capability to create snapshots of detailed watch activity.

For more information about the WebSphere Transformation Extender Launcher Monitor, refer to the *Launcher documentation*.

Snapshot Viewer

The WebSphere Transformation Extender Snapshot Viewer is a GUI that displays snapshots taken in the Launcher Monitor. These snapshots show details about the activity of the Launcher at a specific moment in time.

For more information about the Snapshot Viewer, refer to the Launcher documentation.

B2B management (Trading Manager)

The WebSphere Transformation Extender Trading Manager is an optional B2B management component. It is a client/server product for managing and processing electronic commerce data that provides the capabilities for managing and controlling the Business-to-Business (B2B) integration of partner relationships and message flow. Trading Manager users can audit, control, monitor, and view the entire B2B integration environment across the extended enterprise with secure data exchange fully integrated with back-end systems. The Trading Manager consists of the following components:

- Partner Manager

  Partner Manager is a graphical administration environment used to trade partner profiles and relationships. Partner Manager also provides runtime monitoring, auditing, and reporting functions.
- Message Manager

  Message Manager is a runtime subsystem that uses trading profiles maintained by the Partner Manager to automate document exchange among e-commerce partners.

For more information about Trading Manager and its components, see the Trading Manager documentation.

Partner Manager

The WebSphere Transformation Extender Partner Manager is a GUI application used to create a comprehensive database of an organization's electronic commerce information and to manage its e-commerce activity. Partner Manager is used to:

- define the trading relationships with Electronic Data Integration (EDI) partners, including details such as the version of each document, control numbering schemes, tracking, error handling, status, and post office used.
- maintain the information for each trading partner, including details regarding addresses, contacts, and the individual departments or other groups within trading partner organizations.
- provide audit and control information collected in the Commerce Manager database that is used to monitor ongoing inbound and outbound e-commerce transaction activity.
- provide alerts when error conditions occur in e-commerce inbound or outbound processing.

For more information about the Partner Manager, refer to the Partner Manager documentation.

Message Manager

The WebSphere Transformation Extender Message Manager is a predefined system of maps specifically designed to process electronic commerce data. The maps are used to validate and route data. The Message Manager is the runtime component of the Trading Manager.

Electronic commerce data flows from external sources to internal destinations, from internal sources to external destinations, and possibly to other internal organizations. The Trading Manager manages this flow and the tracking of e-commerce data.

The Message Manager supports two distinct data flows: inbound and outbound. The inbound data flow typically represents the receipt of data in EDI or proprietary format from an external trading partner. The outbound data flow typically represents the retrieval of data from an internal application system in its native format for subsequent conversion and routing to an external trading partner.

The information from the Partner Manager is shared with the Message Manager for the validation, tracking, and routing of e-commerce data. Information about this e-commerce activity is then incorporated into the Partner Manager database for monitoring and reporting.

Trading Manager can be used for internal/external, external/external, and internal/internal trading scenarios.

For more information about the Message Manager, refer to the Trading Manager documentation.

Other WebSphere Transformation Extender solutions

IBM supplies a range of products that extend the out-of-the-box integration services provided by WebSphere Transformation Extender and the Software Development Kit. There are Industry Solutions and Enterprise Application packages contain predefined interface definitions (type trees), adapters, importers, and other components designed to facilitate integration with various enterprise applications and business-to-business interfaces.

Resource adapters

There are an extensive collection of resource adapters that can be used with WebSphere Transformation Extender and the SSDK to connect WebSphere Transformation Extender maps to middleware, DBMSs, and utility services.

Resource adapters can be used to indirectly integrate with enterprise applications that consume inputs or produce outputs using message queues, standard transports, file systems, and databases. In addition, resource adapters enable message-based integration with external and internal applications, including applications and services from SAP, Ariba, Commerce One, i2, Oracle, and others.

- database adapters

  These are used for integration with DB2, Informix, ODBC, OLE DB, Oracle, Microsoft SQL Server, and Sybase.
- Internet adapters

  These are used for integration using email, FTP and HTTP.
- Message-Oriented Middleware (MOMs) adapters

  These are used for integration using BEA MessageQ, BEA Tuxedo, ROMA BSP, WebSphere MQ, Microsoft MSMQ, Oracle AQ, TIBCO Rendezvous, and TIBCO Rendezvous TX.
- Synchronous adapters

  These are used for integration using COM+ and Sockets.
- Utility adapters

These are used for integration with archive, batch, and directory (LDAP) services.

- File adapters

  These are used for integration with platform-specific file systems.

- In addition to pre-built WebSphere Transformation Extender resource adapters and packs, the WebSphere Transformation Extender Adapter Toolkit provides a standard interface and set of services for implementing your own adapters to connect directly with other applications and services.

For more information about the WebSphere Transformation Extender adapters, refer to the Resource Adapters documentation.

Chapter 7. Integration maps

Maps are at work whether implementing:

- an asynchronous, event driven architecture
- a component or Enterprise JavaBean (EJB) architecture
- a synchronous, web-based architecture
- an application-embedded architecture
- a batch interface architecture

Maps provide a non-programmatic approach to the development of interfaces, with prebuilt functionality for connectivity, transformation, and routing. The benefits of the same components prevailing for all integration models are that:

- Components can be created once and reused many times for different types of integration scenarios.
- The skills required to deploy these different models are consistent and reusable.

Anatomy of a map

A map is made up of the following components:

- Input and output map cards that represent the sources and targets.
- Map rules that define how the output data is built based upon business rules.
- Specifications of the resource adapter used to retrieve and send data.

Input and output cards

Each input and output is represented in a separate map card. The data structure for each of the inputs and outputs is represented in the form of a type tree created using the Type Designer.

Transformation, routing, and business logic rules

Type trees are completely interchangeable between inputs and outputs. The same type tree can be reused multiple times for as many different maps as is required. After the output card(s) for a map has/have been associated with a type in a type tree in the Map Designer, a rule cell for each object within the output card object is available. Rules are constructed in these cells, defining how the output data will be built. The components of map rules are color-coded to assist in reading them.

Adapter configuration for input and output

WebSphere Transformation Extender resource adapters provide the connectivity needed to retrieve data from sources (using the `GET` setting) and send data to targets (using the `PUT` setting). Adapters do not provide transformation. That functionality is provided by the map rules. Map rules describe the nature of the transformation. The transformation server performs the actual transformation of the data.

WebSphere Transformation Extender includes adapters for common, enterprise-level resources. In its simplest form, specifying an adapter for an input or output card means selecting the required adapter from a list.

Adapter settings defined in the map can be overridden with different adapter settings at map run time.

The resource adapters are loosely coupled with the transformation process. An adapter retrieves data to be processed by the map as a byte stream. This means that, to the extent that the type definitions and byte streams are compatible, type trees and adapters can be used interchangeably.

For example, when testing, it may be faster to debug problems in the logic of the map using a file. However, in the production environment, the data source may be from a database. As long as the format of the data retrieved from the database is consistent with the format of the data in the file that was used for development and debugging, the only change that has to be made to the map is to change the relevant input(s) from using the **File** adapter to using the **Database** adapter.

Furthermore, the map source file does not need to be changed at all because the map execution settings, including the adapter configuration settings, can be overridden at run time.

Where possible, resource adapters use the native client libraries of the concerned resources to connect to those resources. It is through this mechanism that WebSphere Transformation Extender products can achieve ″direct″ connectivity to those resources. For example, the WebSphere Transformation Extender Oracle adapter uses the Oracle Net8 client to achieve connectivity to Oracle databases.

WebSphere Transformation Extender provides an open API to allow for the development of a user-defined adapter, should the need arise. This allows connectivity to unforeseen, possibly proprietary systems (such as a ″home-grown″ HR system), in addition to the standards-based adapters that WebSphere Transformation Extender provides.

For more information about custom adapters, refer to ″Application-Embedded Execution Model″.

Map behavior during execution

The execution characteristics of a map are determined by the execution parameters that are configured at design time. However, these map execution characteristics or map settings can also be overridden at run time.

The behavior of a single map invocation is as follows:

1. The adapters that are configured for their respective input cards retrieve some of the input data.

   The input data is validated against the type tree definition of the respective input card(s). This ensures that only valid data is passed to the output building process. It is for this reason that *any* piece of *any* of the input data can be used to build *any* of the output data. All of the input data is validated and stored in the map's workspace before any output data is built. Because the entire workspace is available to the map at the time it has to build the output, any portion of the input data can be used, counted, or manipulated to generate any part of the output data.
2. After the input data has been validated, output objects can be used as inputs for a subsequent output.

You can create temporary outputs to signify map logic in certain situations. The Sink adapter can be employed to ensure that the data is not retained.

By default, the rollback behavior applies to the entire map thread level. Therefore, nothing will be committed to output nor deleted from input until all inputs have been successfully validated *and* all outputs have been successfully written. Many commit and rollback options are available within the configuration settings of each input and output card. By manipulating these settings, you can significantly change the default commit and rollback behavior of a map.

Input and output card settings

Card settings define the data object that the card represents and how the data is retrieved or routed and compiled into the map. The behavior of input and output cards configured in the compiled map can be overridden.

Schema setting

This setting defines the type tree definition used to represent the data source for an input card and the output data structure for an output card.

SourceRule settings

`SourceRule` settings in the input card specify the data to be retrieved and how to do it, where to get it, how much of it to get, and what to do when an error occurs.

FetchAs setting

The `FetchAs` setting applies only to input cards. It indicates how the adapter retrieves the input data. By default, this setting is `Integral`, meaning that all of the input data is retrieved once from the source.

Setting this parameter to `Burst` means that the adapter will retrieve the input data in increments specified in the `FetchUnit` parameter. The units of data correspond to the highest-level repeating object in the type tree. For example, a payroll file contains 105 records. Using `Burst` mode with the `FetchUnit` parameter set to `10`, data will be sent in 11 ″chunks″. The first ten chunks will contain ten records each and the eleventh chunk will contain the remaining five records.

Despite the fact that bursts process some of the map data and that multiple sets of outputs are built, transactional integrity is not sacrificed. If desired, the `Scope` card setting can be set to `Map` to prevent any changes from being committed until *all* bursts have successfully completed.

There are several potential benefits to using `Burst` mode:

- A single map consumes a large quantity of input data.

  However, only a small portion of that data is required to build an individual output. `Burst` mode can be used to minimize the size of the workspace (an internal index of types present in the data stream). In some cases, a smaller workspace can increase map performance.
- Coordinate multiple inputs together.

  For example, the first unit of data for Input Card 1 must be coordinated with the first unit of data for Input Card 2, and so on. If `Burst` mode is not enabled, one input must then act as a ″driver″ that must search for matching outputs.
- Process a single logical unit of work at one time.

For example, you may want to process one row from a database table at one time or one message from a message queue at one time. This would equate to the `Burst` mode being enabled with the `FetchUnit` parameter set to `1`.

TargetRule settings

`TargetRule` settings in the output card specify the data that is output, how the output data is routed, and what to do when an error occurs.

Map settings

The previous section outlined the settings that apply to individual input and output cards. However, there are also map settings that dictate the behavior of the entire map-across *all* input and output cards.

The following map settings provide the ability to control aspects of the map's execution such as:

- `MapAudit`

  This setting specifies options to create an audit of the map's execution, data, and execution settings, in addition to the level of detail of the audit.

- `MapTrace`

  For troubleshooting purposes during development, this setting specifies options to create a trace and the level of detail of the trace.

- `WorkSpace`

  This setting specifies the location (in a file or in memory) of the workspace and the amount of memory to allocate for the execution of the map.

- `Retry`

  If compiled map files, work files, audit log files, or trace files that are needed during map execution cannot be opened, this setting specifies whether to attempt to run the map again and the interval of time between each attempt.

  Almost all of the source, target, and map setting parameters can be overridden with different values at run time. This capability is available for all modes of map execution.

  This allows you to change map execution parameters for different deployment environments, sources, and destinations (such as development, user acceptance testing (UAT), and production environments) without having to modify and recompile the maps.

Resource Registry

The Resource Registry is an application that provides a resource name alias repository that can be used to abstract various parameter settings using aliases that resolve to specific resources within the enterprise. The premise behind the Resource Registry is that it permits generic resource variables to be used within WebSphere Transformation Extender components. These resource variables are then automatically resolved to the actual resources, such as directory paths, database names, message queue names, and server names. Resource variables make it possible to use the same map and system definitions in different deployment environments.

The Resource Registry provides a Java-based graphical user interface (GUI) for creating and managing these resource definitions.

Chapter 8. Execution methods

The core execution engine in WebSphere Transformation Extender products is developed in ANSI C for platform portability and high performance. However, there are a variety of mechanisms that can be used to run integration components, including a Java API that enables Java-based interfaces to maps. Other Java-based mechanisms for map execution are servlets and Enterprise JavaBeans (EJB).

The main methods for map execution are as follows:

Execution Method
Description

command-driven
This is primarily geared towards batch processing using the WebSphere Transformation Extender Command Server. For more information about this method, refer to ″Command-Driven Execution Model″ .

event-driven
This is a method in which systems of maps can be coordinated together by the WebSphere Transformation Extender Launcher to provide a scalable, distributed, multi-threaded, execution environment. For more information about this method, refer to ″Event-Driven Execution Model″ .

application-embedded
This is a method in which maps can be embedded in third-party C, Java, COM, and CORBA applications. When used with an application server or Web server, the components in the Software Development Kit can be used to facilitate web-based integration scenarios. For more information about this method, refer to ″Application-Embedded Execution Model″ .

Command-driven execution model

The WebSphere Transformation Extender Command Server provides a command-driven capability for the execution of integration maps. This model is normally associated with batch-driven, on-demand integration processes. The methods for executing maps in command mode include the following:

executing a single map from a command prompt

- executing multiple maps in sequence using a command parameter file
- executing from the Windows-based Command Server, which provides a graphical user interface (GUI) for loading and running maps, as well as for overriding map settings
- executing as part of a batch file or shell script
- executing as part of a mainframe job as specified in the Job Control Language (JCL)
- executing using commands specified in a scheduling program

As mentioned earlier, by default, the execution settings of a map (such as `MapAudit` and `WorkSpace`) are compiled into the run-time object. In all modes of execution, it is possible to override almost all of these settings at run-time. For Command Server invocation, the overrides are supplied as parameters on the command line.

Execution command overriding a database target example

Suppose you want to run the map **multitran.hp**, overriding some of the settings for the fourth output card, which is a database target.

Use the following command:

```
/install_dir/bin/dtx /install_dir/maps/multitran.hp -G
-OD4R5:3B `-TRACE ERROR'
```

Option	Description	
multitran.hp	This specifies the name of the map to be executed.	
-G	All item properties (including size, presentation, and restrictions) are fully validated.	
-OD4R5:3B `...'	This overrides the default settings compiled into the map using the database target for the data of output card #4 (-OD4) as follows:	
	R5:3	If the database resource is unavailable, attempt to make the connection five more times at three-second intervals.
	B	If the map does not complete successfully, roll back any changes made to the database table.
	`-TRACE ERROR'	Generates an adapter trace file containing error information only.

For more information about execution commands, refer to the Execution Commands documentation.

Event-driven execution model

The event-driven execution model relies upon the Launcher to coordinate the triggering of map threads based upon either input event or time event triggers. With its ability to define complex event-triggering criteria, combined with a robust set of adapters that provide listener interfaces to detect these events, the Launcher is the cornerstone for providing real-time, event-driven integration.

Launcher

The WebSphere Transformation Extender Launcher provides a real-time, event-driven model for executing maps and systems of maps defined in the Integration Flow Designer. The Launcher runs on a server platform and manages one or more systems concurrently. Systems are asynchronous, multi-threaded processes, whose atomic units are transactional maps. Systems can execute in a distributed environment in which an Launcher resides on each server. When events occur, the Launcher is notified and starts maps based upon that notification. Distributed data is received or routed during map execution.

Triggers and watches: asynchronous communication

The set of events that initiate a map within a system is called a *watch*. Each event that contributed to the initiation of a map is called a *trigger*. Triggers are asynchronous events resulting from time or source state changes (for example, the creation of a file in a directory or a message appearing on a queue). The Launcher starts maps based upon triggers defined in the Integration Flow Designer. In some cases, there may be multiple triggers that must collectively exist before a map is initiated. The Launcher manages the coordination of these events to ensure that the correct set of circumstances has occurred before a map is initiated. Although each instance of a map is considered a separate watch, the Launcher must also manage the resource interfaces used by other maps and other instances of the same map to properly synchronize these asynchronous events.

Data sources using database adapters can serve as input event triggers that are defined in the Database Interface Designer, enabled in the Integration Flow Designer, and executed by a Launcher.

The listener interface, supported by various adapters, handles triggering through database updates and messages arriving on queues. Additionally, the Launcher Agent (Windows and UNIX) receives external web requests (high-volume capability) through the Launcher without the need for a separate web server.

Synchronous coordination

Synchronous interaction between steps within the data integration process may sometimes be required. For example, it might be important to complete a transaction only when an acknowledgment of an action is received. Synchronous interaction can be accomplished in different ways:

- Communicate with other maps from a map rule during the data transformation process. Using a map rule in this way, the Launcher manages coordination among maps to avoid deadlocked situations and to ensure that shared resources are reliably accessed or deleted.
- Synchronize coordination to interact with other applications using a resource adapter for a data source or target. For example, when a file is sent to a server using the FTP target adapter, the FTP server response is received before the mapping process completes. When using resource adapters, recovery procedures can be configured for errors that occur during communication. For example, in a scenario in which the Value-Added Network (VAN) source adapter is used to retrieve EDI data from a VAN, the user can specify retry sessions to recover from the error that occurs if the line is disconnected during transmission.

Launcher Architecture

The Launcher architecture is an event-driven software model executing maps. Systems contain maps supporting a system data flow that is specified by using the Integration Flow Designer.

An Launcher runs on a server platform and can concurrently manage one or more systems. IBM WebSphere DataStage TX Launcher functionality is based upon the following premises:

- The event manager sub-component of the IBM WebSphere DataStage TX Launcher controls the initiation of maps based upon the sources to which a map subscribes.
- The resource manager sub-component of the IBM WebSphere Launcher synchronizes shared data and timing interfaces among heterogeneous sources and targets.

- Maps publish data that result from the content transformation of source data.
- Maps are transactional processes, with each map having its own error detection and recovery procedures.
- The Launcher runs as a single process with multiple threads. An associated process monitors input source resource changes.

Launcher methodology

Whenever the Launcher detects a new event trigger, the watches associated with that trigger are either:

- initiated, thereby causing the map to execute
- placed in an initiation pending state, awaiting the completion of all specified triggers
- maintained in an initiation-pending state until operating system resources (such as threads) are available

The Launcher initiates an individual map thread for each successful combination of specified triggers. Multiple instances of the same map component can be invoked and executed simultaneously if the Launcher is configured to take advantage of concurrent multi-threading of map execution for performance. Alternatively, if strictly sequential execution architecture is required (FIFO); configuring the Launcher accordingly can accommodate this. This concurrent-versus-sequential capability can be configured at the individual map component level. As a result, maps contained in a system could be concurrent, while others could be set to execute sequentially.

Business Advantages of the Launcher

Some of the main business advantages to using the Launcher are:

- automatic system execution
- multi-threaded environment for multiple transactions, which allows linear scaling across multiple CPUs
- process prioritization

Audit and transaction logging

The totally flexible architecture of WebSphere Transformation Extender products allows the design and development of sophisticated transaction and audit logging using standard functionality. This is supported by the complete range of resource adapters and the many tools available in the complete line of WebSphere Transformation Extender products.

Maps are transactional processes and each map can be configured to have its own error detection and recovery procedures, as well as auditing capability. The resulting audit logs themselves can be mapped, as input data, to dedicated execution logging, error logging, and exception handling routines.

Typically, audit and transaction logging processing is achieved by designing reusable map systems to process audit logs resulting from map execution.

Launcher Agent

The WebSphere Transformation Extender Launcher Agent, available for both the Windows and UNIX systems, is a light-footprint solution designed to receive external Web requests (high-volume capable) through the Launcher. The Launcher Agent is a utility that listens for incoming HTTP requests and passes them onto the

WebSphere Transformation Extender Launcher to trigger coordinated systems of maps. The difference between handling HTTP requests using the Launcher Agent as compared to a servlet is that the Launcher Agent can process an entire asynchronous system of maps with a single HTTP request. Unlike the servlet, which is essentially one map invocation per servlet (although there are ways to call additional maps in the same session), the Launcher Agent can trigger many maps in a coordinated fashion using the Launcher. Also, unlike the servlet in which the map receiving the initial HTTP request must also be the map sending the resultant HTTP response, the Launcher Agent and Launcher-based invocation permits *any* map in the triggered system (or associated systems) to send the HTTP response to the Launcher Agent and forward it onto the point of origin.

This capability not only allows *conventional* HTTP processing scenarios to be implemented (such as browser-based access to enterprise resources using the Internet or Intranet), but it also supports a growing trend for using HTTP as the primary transport for intra-enterprise application-to-application (A2A) and inter-enterprise business-to-business (B2B) integration. The IBM Launcher Agent works with the HTTP adapter to enable the Launcher to directly process HTTP input events without compromising the integrity of the firewall. The Launcher Agent can be deployed either outside the corporate firewall in the demilitarized zone (DMZ) for business-to-consumer (B2C) or B2B integration, inside the firewall for A2A integration, or in both places.

Because an application or Web server is not required, complete A2A, B2B, and B2C capabilities are provided without the need to resort to programming.

With the installation of the Secure Adapter Collection, which is an option for WebSphere Transformation Extender, the Launcher Agent supports the following protocols:

- Transport layer security
- Secure Sockets Layer (SSL) security

Using the Launcher Agent

The following scenarios describe when to use the Launcher Agent:

- when the HTTP request requires a complex system of maps to generate the response
- if the integration requires the usage of asynchronous messaging
- if a light-footprint solution is required to process internal and external HTTP requests

Business advantages of the Launcher Agent

The following is a list of business advantages gained by using the Launcher Agent:

- single architecture for A2A, B2C, and B2B integration
  - greater re-usability and portability of skills
  - no requirement for any procedural programming
  - ease of ongoing maintenance
  - quicker implementation times
- support on major platforms
  - ability to leverage existing IT investments (technology and skills)
  - cost reduction-no new skill sets needed
- firewall integrity maintained-no requirement to ″punch a hole″ into the firewall from outside while in the DMZ

– business security threats minimized

Application-embedded execution model

The mechanism that is used for this model is basically the same as the one providing run-time instructions for a map's execution from the command prompt. Overrides are submitted to control the map's execution. The difference is that these overrides are programmatically supplied, in memory, rather than on the command line. However, even under WebSphere Transformation Extender Command Server execution, the overrides specified on the command line are passed by the Command Server, in memory, to the core execution code, which is embodied in the various APIs. Note that, regardless of the execution model employed, the same underlying WebSphere Transformation Extender technology prevails.

The benefit of embedding a map within an application is that all of the power of the standardized transformation and connectivity capabilities of the WebSphere Transformation Extender products can be fully leveraged. Rather than needing to write Java, C, or C# code to perform complex transformation, connectivity, and routing, maps can be fully leveraged to perform these tasks. Again, as with other execution models, the same inherent WebSphere Transformation Extender technology components are used: maps. This means that a more unified skill set is applicable across all integration models, thereby reducing the overhead of writing the code to perform transformation, routing, and connectivity to a minimum. This allows more time to concentrate efforts upon developing the business logic in the application layer code, resulting in a much quicker implementation. In addition, because the source components, such as type trees and map source files, are not procedural code listings, but rather visual, object-based components, the ongoing maintenance of these components is more straightforward, using the extensive capabilities provided by the Design Studio.

APIs are provided for the following applications in the SSDK, along with a servlet-based model:

- "C applications"
- "Java applications"
- "COM applications" on page 37
- "CORBA applications" on page 37
- "C# applications" on page 38
- "J2EE applications" on page 38
- "Servlets" on page 38

C applications

The Software Development Kit includes two APIs available to C programmers: the TX Programming Interface and the Platform API. Customers creating new applications should use the TX Programming Interface, C API. The Platform API continues to be supported for legacy applications. However, any new features or functionality will be available only in the TX Programming Interface, C API.

Java applications

There are several ways in which a Java application can run a map:

- By using the TX Programming Interface Java API
- By using the TX Programming Interface RMI API

  - With the RMI API, a map (and the underlying native core code) can be executed out of process. Also, with RMI, WebSphere Transformation Extender can run on any platform that supports the Java RMI API. (And this is not necessarily the same platform as the ones upon which Java Application Servers run.)
  - This enables maps to be executed by an Enterprise JavaBean containing business logic and subsequently to be deployed to an application server. Examples of this are provided with the WebSphere Transformation Extender SSDK.
- By using the TX Programming Interface, CORBA API

  For more information about this, refer to "CORBA applications".
- By using the JCA Connector
  - The JCA Connector provides a standard JCA 1.0 interface for executing maps, facilitating easy integration for JCA-compliant applications.
- By using the EJB API
  - This was supported for previously written applications. However, any new features or functionality will be available only in the RMI API.
  - The EJB API consists of a stateful and stateless Session bean that wraps a map.

Java-based applications that invoke a map through the API can also take advantage of using the Java Class Adapter to access additional Java objects. For EJB functionality, access to serializable objects is supported. From the user's Java Application or EJB, objects are serialized using Java Serialization and passed (using a STREAM adapter override) to input cards of a map. Within the map, the Java Class adapter can be used to deserialize an object (and store it in the Object Pool). Access to fields and methods on Objects stored in the Object Pool is permitted using the Java Class adapter. (Objects are also accessible using the CORBA, JNDI, and JMS adapters.) To pass objects back to the user's application from a map, objects are serialized using the Java Class adapter. The resulting byte[] is passed back to the calling EJB using a STREAM adapter override. The Java application or EJB re-constructs the Object using Java Serialization.

COM applications

For customers using Microsoft's Component Object Model (COM) technology, the WebSphere Transformation Extender Programming Interface (TXPI) for COM provides the functionality to run an WebSphere Transformation Extender map from scripting languages such as VBScript and Jscript. This also allows WebSphere Transformation Extender maps to be executed from Active Server Pages.

CORBA applications

The Software Development Kit (SSDK) also provides a TX Programming Interface for CORBA. CORBA is the acronym for Common Object Request Broker Architecture-OMG's open, vendor-independent architecture and infrastructure. Using the standard IIOP protocol, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, or network can interoperate with a CORBA-based program from the same or another vendor on almost any other computer, operating system, programming language, or network.

The TX Programming Interface for CORBA allows WebSphere Transformation Extender clients to run maps from various languages such as C++, VBScript, Python, Perl, Visual Basic, Java, and so on.

C# applications

Using the TX Programming Interface, COM API, and the interoperability library provided in the SSDK examples, a C# application running on .NET can also run an WebSphere Transformation Extender map for transformation.

J2EE applications

Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing component-based, multi-tier, enterprise applications. Applications built using J2EE can use the various Java-based APIs described.

Servlets

WebSphere Transformation Extender offers a servlet-based execution model that enables a map to be run by a Java servlet. This is particularly useful for Web browser-based interfaces requiring a request and response mode of operation. The map is able to function in a stateful fashion because of the servlet's ability to maintain state on the HTTP session.

New applications should use the capabilities provided by the Transformation Extender Programming Interface for Java and RMI to implement Servlet functionality. The Servlet Integrator is supported for existing applications.

Using the Servlet Integrator, you can automatically generate the necessary servlet code that will interact with the corresponding map. The Servlet Integrator provides the ability to create browser-based request and response interfaces without having to write any procedural code. The servlet generated by the Servlet Integrator utilizes the Java API, which in turn instantiates the maps using the native Platform API function libraries.

Chapter 9. Other features of WebSphere Transformation Extender products

For support, training, or other product information, go to www.ibm.com/software/integration/wtx.

Examples

Another source for learning about WebSphere Transformation Extender products is by running the examples that are provided with the product. The examples illustrate how to use WebSphere Transformation Extender product components to accomplish specific tasks and can prompt ideas for achieving your goals. The examples are located at *install_dir*/examples.

Tivoli License Manager

IBM Tivoli License Manager (ITLM) enables you to monitor the usage of IBM (and other) software products. ITLM provides you with the following software auditing functionality:

- Monitor the licenses used across different machines.
- Help keep unnecessary licenses to a minimal.
- Guard against software license compliance problems.

Version 8.1 of WebSphere Transformation Extender includes support for ITLM V2.1. To find out more about using ITLM to monitor usage of your IBM software, see the IBM Tivoli License Manager Web site.

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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

AIX
AIX 5L
AS/400
Ascential
Ascential DataStage
Ascential Enterprise Integration Suite
Ascential QualityStage
Ascential RTI
Ascential Software
Ascential
CICS
DataStage
DB2
DB2 Universal Database
developerWorks
Footprint
Hiperspace
IBM
the IBM logo
ibm.com
IMS
Informix
Lotus
Lotus Notes
MQSeries
MVS
OS/390
OS/400
Passport Advantage
Redbooks
RISC System/6000
Roma
S/390
System z
Trading Partner
Tivoli

WebSphere
z/Architecture
z/OS
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation 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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Transformation Extender, Version 8.1

Index

K

L

M

O

P

R

S

T

W

X

Printed in USA