IBM InfoSphere Information Services Director
Version 9 Release 1

User's Guide
Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page [8].
Index .................................................. 87
Chapter 1. Introduction to IBM InfoSphere Information Services Director

You can more easily create business processes with information from a complex, heterogeneous environment if that information is published as consistent and reusable services.

Information integration is a key part of creating services with IBM® InfoSphere® Information Server. IBM InfoSphere Information Services Director allows units from any of the suite components to be deployed as Web services or Enterprise Java™ Beans (EJBs) in minutes. IBM InfoSphere Information Services Director load balances service requests across multiple InfoSphere Information Server nodes, to ensure smooth pickup of load spikes, and to ensure fault tolerance and high availability. It provides the following key capabilities:

- Packaging information integration logic as services that insulate developers from underlying sources
- Allowing these services to be invoked as EJBs or Web services
- Provides REST access to services using the XML or JSON format
- Exposing services as RSS feeds
- Using the JMS transport method for asynchronous access to service responses
- Providing load balancing and fault tolerance for requests across multiple servers
- Providing foundation infrastructure for information services

The extensible architecture of InfoSphere Information Services Director allows it to enable a broad range of information management tasks such as data cleansing, data transformation, and data federation services.

Infrastructure services

InfoSphere Information Services Director is deployed on a J2EE-based service backbone that provides flexible, distributable and configurable interconnections among the many parts of the architecture. These infrastructure services include:

Logging services
Provide a central place for a user to record service events. Logs go into the common repository with each service provider defining relevant logging categories for that service. Configurations determine the categories of logging messages that are saved in the repository.

Security services
Support role-based authentication of users, access-control services and encryption appropriate for compliance with many privacy and security regulations.

Service catalog
Provides users with the means to search and browse services by category and to view descriptions that can be defined when the service is created. InfoSphere Information Services Director also provides a direct link to the IBM WebSphere® Service Registry Repository for those who have this product.

Load balancing and availability services
Support routing requests to multiple servers to provide optimal loading and a high availability environment that can recover from any individual server failure.
IBM InfoSphere Information Services Director and IBM InfoSphere
Information Server architecture

To understand how IBM InfoSphere Information Services Director works, you must understand the architecture of IBM InfoSphere Information Server.

As shown in the following figure, InfoSphere Information Server is installed in four tiers: client, metadata repository, services, and engine. Product components are installed in each tier depending on the selections that you make. The tiers can be installed on the same computer or on different computers. You should understand where each of the tiers is installed and how they impact installation. The following figure shows the client, documentation, metadata repository, services tier, and engine installed across two computers. Computer A is a Microsoft Windows XP or Windows Vista system and Computer B is a Windows Server 2003, Linux, or UNIX system.

Figure 1. InfoSphere Information Server configuration

The services tier

InfoSphere Information Services Director server resides in the services tier of InfoSphere Information Server. The services tier is contained on an IBM WebSphere Application Server. InfoSphere Information Services Director itself is a set of Enterprise Java Beans (EJB) that runs in the application server. Every InfoSphere Information Server service generated by InfoSphere Information Services Director is generated
as an EJB session bean regardless of whether the function is an IBM InfoSphere DataStage® job, IBM InfoSphere QualityStage® job, database-stored procedure, or federated query. After the service session bean has been generated, additional artifacts get created depending upon whether SOAP over HTTP or EJB binding is requested for the service. The services tier is also where the IBM InfoSphere Metadata Server programs reside, which provide InfoSphere Information Services Director with the IBM InfoSphere Metadata Server services that create and store InfoSphere Information Services DirectorInfoSphere Information Services Director runtime metadata.

The client tier

You develop, deploy, and maintain information services through two client applications:

**IBM InfoSphere Information Server console**

The console is a task-oriented user interface that integrates suite components into one unified framework. The console contains workspaces that you use to create and complete information integration tasks such as investigating data, creating job schedules and logs, and deploying applications or Web services.

**IBM InfoSphere Information Server Web console**

The Information Services Catalog is part of the IBM InfoSphere Information Server Web console. The Information Services Catalog contains information about deployed information services. It allows you to browse the services categories tree in the catalog and search categorized, uncategorized deployed services, or all services by custom attributes and design-time metadata.

The engine tier

The engine tier is where much of the work performed by InfoSphere Information Server takes place. For InfoSphere Information Services Director, the engine tier is where the actual information providers reside (although not all information providers need to be part of InfoSphere Information Server). InfoSphere DataStage, InfoSphere QualityStage and IBM InfoSphere Federation Server are all parts of the engine tier. When InfoSphere DataStage or InfoSphere QualityStage servers are installed they are registered to a particular domain. The InfoSphere Information Services Director server can only connect to those servers that are registered to the same domain as the one in which the InfoSphere Information Services Director server resides. This is because InfoSphere DataStage and InfoSphere QualityStage share the common services such as security as InfoSphere Information Services Director and therefore they must reside in the same domain. Information providers such as IBM InfoSphere Federation Server and IBM DB2 are not associated with a specific domain and therefore do not have this restriction.

**IBM InfoSphere Information Services Director message flow**

IBM InfoSphere Information Services Director service requests take a series of steps to be processed.

The following figure shows a single InfoSphere Information Services Director server and two Application Service Backbone (ASB) Agents. The InfoSphere Information Services Director endpoints can be for SOAP over HTTP, EJB, and so on. When a request arrives at the endpoint, it is passed to the Application Service Backbone (ASB) Adapter along with information about the requested operation. The operation metadata is used to determine the set of providers that can process the request, and the load balancer to be used to pass the request to the correct ASB Agent. The ASB Agent is the part of the Information Service Framework (ISF) that provides a means of sending requests from the ASB Server (where IBM InfoSphere Information Server is installed) to remote systems where the ASB Agent is located without the need for a full J2EE Application Server at each client location. The ASB Agent utilizes a plug-in architecture to allow different types of requests to be passed from the ASB Server.
The Information Service Framework also provides load balancing and pooling of resources. The load balancer is only used to balance the load between two different ASB Agents. Therefore, if an operation has only one provider attached to it, or if all attached providers run on the same ASB Agent, then no load balancing will be done. The load balancer passes the request to the ASB Agent where it is placed in a queue for the operation that is being processed. When a slot becomes available in a pipeline, the request will move from the Operation Queue to the pipeline where it is processed by the corresponding IBM InfoSphere DataStage job, IBM InfoSphere Federation Server or IBM DB2 query, stored procedure, or other provider. At runtime the user can affect the behavior of the load balancer and these queues.

**Related concepts:**
- "Runtime settings" on page 66

During the design of the service operation, default settings for runtime execution are created.
Chapter 2. Supported information service providers

Information service providers are the sources of operations for your services. Using IBM InfoSphere Information Services Director, you can create services from the following information service providers: IBM InfoSphere DataStage; IBM InfoSphere QualityStage; IBM DB2 for Linux, UNIX, and Microsoft Windows; IBM InfoSphere Federation Server; IBM InfoSphere Classic Federation Server for z/OS®; and Oracle Database Server.

About this task

To use an information service provider as a source for InfoSphere Information Services Director, you must be able to connect to the service provider from the IBM InfoSphere Information Server console.

Table 1. Information service providers supported by InfoSphere Information Services Director

<table>
<thead>
<tr>
<th>Information service provider</th>
<th>Operations that you can expose by using the IBM InfoSphere Information Server console</th>
</tr>
</thead>
<tbody>
<tr>
<td>InfoSphere DataStage or InfoSphere QualityStage</td>
<td>InfoSphere DataStage and InfoSphere QualityStage jobs</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>Queries or stored procedures from IBM DB2 or SQL queries that you create by using the SQL builder that is available within the console</td>
</tr>
<tr>
<td>IBM InfoSphere Federation Server</td>
<td>Queries or stored procedures from IBM InfoSphere Federation Server or SQL queries that you create by using the SQL builder that is available within the console</td>
</tr>
<tr>
<td>IBM InfoSphere Classic Federation Server for z/OS</td>
<td>Queries or stored procedures from IBM InfoSphere Classic Federation Server for z/OS or SQL queries that you create by using the SQL builder that is available within the console</td>
</tr>
<tr>
<td>Oracle Database Server</td>
<td>Queries or stored procedures from Oracle Database Server or SQL queries that you create by using the SQL builder that is available within the console</td>
</tr>
</tbody>
</table>

Related concepts:

Chapter 6, “Developing applications, services, and operations,” on page 29

As shown in the following figure, the IBM InfoSphere Information Services Director development model is based on a hierarchy of objects that define the flow of development.

Related tasks:

“Adding operations” on page 57

Add operations to define the information tasks of a service. Each information service is composed of a set of operations that use artifacts such as database queries for its business logic. The implementation of operations, such as database queries and IBM InfoSphere DataStage jobs, is hidden from the service client. This loose coupling enables service clients to be developed independently of information services, and requires minimal coordination between the service developer and the service client developer.
Chapter 3. Designing IBM InfoSphere DataStage and QualityStage jobs as services

Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job topologies for different load and work style requirements: batch jobs, batch jobs with a service output stage, and jobs with service input and output stages.

The design of a job determines whether it is always up and running or runs once to completion. All jobs that are exposed as services process requests on an ad-hoc, 24x7 basis. The IBM InfoSphere Information Services Director server starts job instances on one or more InfoSphere DataStage servers for load balancing and scalability.

You must specify whether a job is always running or enabled for service in the job properties window of the IBM InfoSphere DataStage and QualityStage Designer.

Make sure the column names in your InfoSphere DataStage and IBM InfoSphere QualityStage jobs do not contain a numeric digit followed by lower case letter, such as 2b. This type of column name will cause your jobs to fail when you invoke them as IBM InfoSphere Information Services Director services.

A service accepts requests from client applications, mapping request data to input rows and passing them to the underlying jobs. A job instance can include database lookups, transformations, data standardization and matching, and other data integration tasks. A job instance can then return output rows that can be mapped to service response data and sent back to the client.

Batch jobs

Topology I uses new or existing batch jobs that are exposed as services. A batch job starts on demand. Each service request starts one instance of the job that runs to completion. This job typically initiates a batch process from a real-time process that does not need direct feedback on the results. It is tailored for processing bulk data sets and is capable of accepting job parameters as input arguments. Topology I jobs have the following characteristics:

Start and stop times
   The elapsed time for starting and stopping a batch job, also known as latency, is high. This factor contributes to a low throughput rate in communication with the service client.

Job instances
   The Information Service Framework (ISF) agent starts job instances on demand to process service requests, up to a maximum that you configure. For load balancing, you can run the jobs on multiple InfoSphere DataStage servers.

Input and output
   An information service that is based on a batch job can use job parameters as input arguments. This type of service returns no output. If you design the information service, you can set values for job parameters. If the job ends abnormally, the service client receives an exception.

Batch jobs with a service output stage

Topology II uses an existing batch job and adds an output stage. The InfoSphere Information Services Director output stage is the exit point from the job, returning one or more rows to the client application as a service response. Its table definition maps to the output arguments of a service operation, such as the return value of a Web service operation. Return values can consist of an atomic value (one column), a structure (multiple columns), or an array of structures (multiple rows). As the following figure shows,
these jobs typically initiate a batch process from a real-time process that requires feedback or data from the results. It is designed to process large data sets and is capable of accepting job parameters as input arguments. Requirements for a Topology II job are identical to those for Topology I jobs in all other respects.

**Figure 3. Topology II job**

### Jobs with service input and output stages

In Topology III, jobs use both an InfoSphere Information Services Director input stage and a service output stage. The input stage is the entry point to a job, accepting one or more rows during a service request. These jobs are always running. This topology is typically used to process high volumes of smaller transactions where response time is important. It is tailored to process many small requests rather than a few large requests. The following figure shows an example of this topology.
Topology III jobs have the following characteristics:

**InfoSphere Information Services Director input**

The InfoSphere Information Services Director input stage supports one output link. Its table definition maps to the input arguments of a service operation, such as the input arguments of an EJB method. Input values can consist of an atomic value (one or more columns), a structure (multiple columns), or an array of structures (multiple rows).

**Defining stages**

Jobs with passive stages that have both input and output links are not eligible to implement service operations. The job that has a data source stage with input and output links cannot be used to implement a service operation, because the data source stage acts as a synchronization point. If you need a data source stage with both input and output links, do not use an InfoSphere Information Services Director input stage. If you omit the InfoSphere Information Services Director input stage from a job, more processing time for a service request is needed, including the time to start and stop the job.

Also, a database query with results that are returned on an output link is run only one time, when the job instance is started. It is not re-run during every service call request. If you want to extract data from a data source, use a reference link to perform the lookup.

**Always-on behavior**

A job that conforms to Topology III is always running. If you design the service operation, you select the following options:

- Minimum number of job instances

---

*Figure 4. Topology III job*
• Maximum number of job instances

Each job instance handles multiple requests during its lifetime. The ASB agent starts job instances prior to service requests, which virtually eliminates latency. This factor contributes to a high throughput rate in communication with the service client. If the job stops, the service client receives an exception.

To avoid timeouts among your database connections, set the maximum lifetime of job instances just below the lowest timeout limit. This is a deployment step. At runtime, the ASB agent recycles instances to meet demand and to maintain the minimum required instances.

Job parameters

Any job parameters that belong to a Topology III job are static for all job instances. If you design the service operation, you can set job parameter values.

Reuse considerations

Jobs with an InfoSphere Information Services Director input stage carry design constraints.

Related concepts:

“Global considerations”
This section describes the following global considerations that apply to all three IBM InfoSphere DataStage job topologies.

“Job design and runtime constraints” on page 12
In job design, some features and runtime behaviors apply only to certain IBM InfoSphere DataStage job topologies.

“Data Source stage with input and output links” on page 13
A job that has an ISD Input stage and a data source stage with input and output links does not work because the data source stage acts as a synchronization point.

“Data Source stage with stream output link” on page 13
A database query whose results are returned on an output link is started only one time, when the job instance is started. It is not restarted during every service call request.

“Sequential File stage” on page 13
File names used by the Sequential File stage must be unique for each job instance.

“FTP stage” on page 14
FTP stage destination files must be unique for each job instance.

“Database stages” on page 14
Follow these guidelines when using database stages in a job.

“Always-running instances” on page 14
Follow these guidelines for jobs that are always-running instances.

“About services” on page 34
An information service is a container for a group of related operations. When you create services, you indicate the method that you use to invoke operations within a service. In addition, you indicate your security requirements at the service level.

Related tasks:

“Adding a DataStage and QualityStage type” on page 58
Complete this task to expose an IBM InfoSphere DataStage job or an IBM InfoSphere QualityStage job as a service.

Global considerations

This section describes the following global considerations that apply to all three IBM InfoSphere DataStage job topologies.
Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job topologies for different load and work style requirements: batch jobs, batch jobs with a service output stage, and jobs with service input and output stages.

**Macros and BASIC functions**

Within an IBM InfoSphere DataStage job that is exposed as a service, you can include the following macros and BASIC functions.

This topic uses the following definitions:

- **transmission block**
  A unit of work processed from a stream of data.

- **end-of-transmission (EOT) message**
  A message used to indicate the conclusion of a transmission.

- **end-of-file (EOF) message**
  A message that signals that the last record of a file has been read.

From a conceptual point of view, a job with an ISD Input stage splits data into transmission blocks by using end-of-transmission (EOT) messages. The first transmission block is received when the job starts. The first transmission block ends and the second begins when the first EOT message is received. This process continues until the job ends with an end-of-file (EOF) message.

EOTs flow through the graph of active stages in a job. Each stage keeps status information about the EOTs that it has received. When all of the active job stages process the last EOT and EOF, the EOT status is updated.

The API only provides information about the processing of transmission blocks for server jobs.

**Macros**

- **DSRTIRowCount**
  Indicates the row number within the current transmission block.

- **DSRTIRequestNo**
  Indicates the current transmission block number. The first transmission block number is 1.

**BASIC Functions**

- **DSGetJobInfo**, with the following infotypes:
  - **DSJ.JOBEOTCOUNT**
    This key returns a count of EOTs that have been processed completely by the job.
  - **DSJ.JOBEOTTIMESTAMP**
    This key returns a time stamp that marks the time at which the last transmission block was processed. The time stamp is in the form of YYYY-MM-DD HH:MM:SS:mmm. (YYYY equals year, MM equals month, DD equals day, HH equals hour, MM equals minute, SS equals seconds, mmm equals thousandths of a second.)
  - **DSJ.JOBRTISERVICE**
    This key returns the value 1 if the job is enabled to publish as an IBM InfoSphere Information Services Director service.

- **DSGetStageInfo**, with the following infotypes:
DSJ.STAGEEOTCOUNT
This key returns a count of EOTs that have been processed completely by the stage.

DSJ.STAGEEOTTIMESTAMP
This key returns a time stamp that marks the time at which the last transmission block was processed. The time stamp is in the form YYYY-MM-DD HH:MM:SS:mmm. (YYYY equals year, MM equals month, DD equals day, HH equals hour, MM equals minute, SS equals seconds, mmm equals thousandths of a second.)

DSJ.STAGEEOTSTART
This key returns a time stamp that marks the time at which the last transmission block was started. The time stamp is in the form YYYY-MM-DD HH:MM:SS:mmm. (YYYY equals year, MM equals month, DD equals day, HH equals hour, MM equals minute, SS equals seconds, mmm equals thousandths of a second.)

DSGetLinkInfo, with the following infotypes:

DSJ.LINKROWCOUNT
This key returns the total number of rows that have flowed down the link from the time that the job started.

DSJ.INSTROWCOUNT
This key returns one comma-separated list of row counts for each instance of a parallel job.

DSJ.LINKEOTROWCOUNT
This key returns the number of rows that have flowed down the link since the last transmission block was processed.

After-stage routines
When an IBM InfoSphere DataStage job has an ISD Input stage, after-stage routines run only when the job is disabled.

Adding an ISD Input stage to a job means that the job is always running. After-stage routines are executed only when jobs are stopped. Jobs that include the ISD Input stage stop running when you disable them in the IBM InfoSphere Information Services Director console or when provider properties settings such as maximum run time limit the amount of time that a job runs.

Branching in stage flow
In an IBM InfoSphere DataStage job, the stage that is directly upstream of an ISD Output stage can split its output between the ISD Output stage and another stage.

If the job terminates, the ISD Output stage might not be notified. To ensure that the ISD Output stage does not wait, use a Link Collector stage to deliver the results from the branch to the ISD Output stage.

Job design and runtime constraints
In job design, some features and runtime behaviors apply only to certain IBM InfoSphere DataStage job topologies.

This section explores the following design features and runtime behaviors that impose constraints in one or more topologies:

• Data source stage with input and output links
• Data source stage with stream output link
• Multiple job instances
• Always-running instances (jobs that have an ISD Input stage)
Related concepts:

Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7

Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job topologies for different load and work style requirements: batch jobs, batch jobs with a service output stage, and jobs with service input and output stages.

Data Source stage with input and output links

A job that has an ISD Input stage and a data source stage with input and output links does not work because the data source stage acts as a synchronization point.

Note: This design feature of an IBM InfoSphere DataStage job is applicable for Topology III (jobs with ISD Input and ISD Output stages).

The output link passes the workload downstream when all the upstream data is received on the input link.

When you use an ISD Input stage, the input link waits indefinitely for data because the life of a job extends across multiple requests.

If you want to use a data source stage with both input and output links, do not use an ISD Input stage. Dropping the ISD Input stage results in additional processing time for a service request, including the time to start and stop the job.

Related concepts:

Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7

Data Source stage with stream output link

A database query whose results are returned on an output link is started only one time, when the job instance is started. It is not restarted during every service call request.

Note: This design feature of an IBM InfoSphere DataStage job is applicable for Topology III (jobs with ISD Input and ISD Output stages).

To extract data from a data source, use a reference link to perform the lookup.

Related concepts:

Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7

Multiple job instances

Because multiple instances of the same job can run simultaneously depending on the deployment configuration, consider such things as concurrent access to resources like files and databases when you design your job.

Sequential File stage

File names used by the Sequential File stage must be unique for each job instance.
Note: This design feature of an IBM InfoSphere DataStage job is applicable for Topology I (batch jobs),
Topology II (batch jobs with an ISD Output stage), and Topology III (jobs with ISD Input and ISD Output
stages).

All job instances will try to update the same file. To avoid these collisions, use the InfoSphere DataStage
macro $\text{DSJobInvocationId}$ to vary the file name for each job instance. IBM InfoSphere Information Server
creates a unique numeric identifier during a job run that you can access using this macro.

Related concepts:
Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7
Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set
of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job
topologies for different load and work style requirements: batch jobs, batch jobs with a service output
stage, and jobs with service input and output stages.

FTP stage
FTP stage destination files must be unique for each job instance.

Note: This design feature of an IBM InfoSphere DataStage job is applicable for Topology I (batch jobs),
Topology II (batch jobs with an ISD Output stage), and Topology III (jobs with ISD Input and ISD Output
stages).

All job instances will try to update the same file. To avoid these collisions, use the InfoSphere DataStage
macro $\text{DSJobInvocationId}$ to vary the filename each time a file is written.

Related concepts:
Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7
Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set
of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job
topologies for different load and work style requirements: batch jobs, batch jobs with a service output
stage, and jobs with service input and output stages.

Database stages
Follow these guidelines when using database stages in a job.

Note: This design feature of an IBM InfoSphere DataStage job is applicable for Topology I (batch jobs),
Topology II (batch jobs with an ISD Output stage), and Topology III (jobs with ISD Input and ISD Output
stages).

- Do not configure the stage to empty and create tables at the start of a job because IBM InfoSphere
  Information Server will empty a table each time a new instance is started.
- To reduce the likelihood of table locking, use the AutoCommit feature or a Commit frequency of 1. Do
  not configure the stage to commit rows at the end of a job.

Related concepts:
Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7
Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set
of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job
topologies for different load and work style requirements: batch jobs, batch jobs with a service output
stage, and jobs with service input and output stages.

Always-running instances
Follow these guidelines for jobs that are always-running instances.

Note: This design feature of an IBM InfoSphere DataStage job is applicable for Topology III (jobs with
ISD Input and ISD Output stages).
To avoid timeouts among your database connections, set the maximum lifetime of job instances just below the lowest timeout limit. This is a runtime setting that you select during the deployment step in IBM InfoSphere Information Services Director. At runtime, IBM InfoSphere Information Server recycles instances to meet demand and to maintain the minimum required instances.

To reduce the likelihood of table locking, in IBM InfoSphere DataStage and QualityStage Designer select the **Autocommit** feature or a Commit frequency of 1. Do not configure the stage to commit rows at the end of a job.

**Related concepts:**

- Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7
- Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job topologies for different load and work style requirements: batch jobs, batch jobs with a service output stage, and jobs with service input and output stages.

“Runtime settings” on page 66

During the design of the service operation, default settings for runtime execution are created.

---

**Adding the ISD Output stage**

You configure the ISD Output stage while building a job in IBM InfoSphere DataStage and QualityStage Designer.

**About this task**

A job can have one ISD Output stage or no ISD Output stages.

**Procedure**

1. From the **Real Time** group in the **Palette** pane, drag the **ISD Output** stage icon onto the canvas.
2. Connect the **ISD Output** stage icon to the stage that supplies the outgoing data.
3. Double-click the **ISD Output** stage icon to display properties for the stage.
4. Click the **Stage** tab.
5. Click the **Properties** tab.

   **Note:** You cannot use job parameters for buffer size and timeout for the ISD Input stage or the ISD Output stage. Enter fixed values for these job properties when you create a service in IBM InfoSphere Information Services Director, IBM InfoSphere Information Server and the InfoSphere Information Services Director agent must access these values before you start the job.

6. Type a value for **Buffer Size**.
7. Type a value for **Timeout**. The default value of 30 seconds is adequate for most environments.
8. Click the **Inputs** tab.
9. Click the **Columns** tab.
10. Type the column names or click **Load** to automatically add the input column names.

**Related concepts:**

- “Timeout value for ISD Input and Output stages” on page 23

The **Timeout** value is the maximum time that the ISD Input stage or the ISD Output stage waits to communicate with the IBM InfoSphere Information Services Director Agent.

---

**Adding the ISD Input stage**

You configure the ISD Input stage while building a job in IBM InfoSphere DataStage and QualityStage Designer.
### About this task

A job can have one ISD Input stage or no ISD Input stages.

### Procedure

1. From the **Real Time** group in the **Palette** pane, drag the **ISD Input** stage icon onto the canvas.
2. Connect the **ISD Input** stage icon to the stage that receives the incoming data.
3. Double-click the **ISD Input** stage icon to display properties for the stage.
4. Click the **Stage** tab.
5. Click the **Properties** tab.

   **Note:** You cannot use job parameters for buffer size and timeout for the ISD Input stage or the ISD Output stage. Enter fixed values for these job properties when you create a service in IBM InfoSphere Information Services Director. IBM InfoSphere Information Server and the InfoSphere Information Services Director agent must access these values before you start the job.

6. Type a value for **Buffer Size**.
7. Type a value for **Timeout**. The default value of 30 seconds is adequate for most environments.
8. Click the **Outputs** tab.
9. Click the **Columns** tab.
10. Type the column names or click **Load** to automatically add the input column names.

   **Note:** If you plan to use the Text Over JMS service binding, use only one input column.

### Related concepts:

<table>
<thead>
<tr>
<th>Related concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Timeout value for ISD Input and Output stages” on page 23</td>
</tr>
</tbody>
</table>

The **Timeout** value is the maximum time that the ISD Input stage or the ISD Output stage waits to communicate with the IBM InfoSphere Information Services Director Agent.

### Creating real-time jobs with two input sources

Real-time jobs cannot contain more than one ISD Input stage and more than one ISD Output stage. However, several stages require more than one input. In the job design, ensure that the inputs provide data for each request of the service.

### About this task

A **real-time** job is a job that is always running as a service.

One of the main principles in real-time job design is that the ISD Input stage must drive the data through the job flow. If a path in the job has an origin that is independent of the ISD Input stage path, special consideration should be given to the design to prevent unpredictable behavior when a single instance of the job is called by more than one service request.

To design real-time jobs with stages that have two or more input sources, apply the principles that are demonstrated in these examples of Two-source Match jobs. If you do, your job designs will align with the runtime behavior of the ISD Input stage.

The Two-source Match stage requires two input sources. Other examples of stages that require more than one input are the Join, Merge, and Funnel stages.
Design considerations for real-time Two-source Match jobs

A Two-source Match job requires input from a data source and a reference source, which require additional considerations during job design. The examples in this section reflect these considerations in the method of selecting reference records.

The following considerations apply to real-time Two-source Match jobs:

- As with any IBM InfoSphere DataStage and QualityStage job, Two-source Match jobs can contain only one ISD Input stage and only one ISD Output stage. Although the Two-source Match stage requires two inputs, your job must accommodate a single ISD Input stage. The Two-source Match stage also requires two inputs for frequency data from the input and reference data, but the Two-source Match stage handles that data without explicit action.

- If you want to use a dynamic view of your reference data as input to the match, use a database and a Lookup stage. Also, use a processing stage such as the Copy stage to copy data from the ISD Input stage to the Lookup and the Two-source Match stages. In the Lookup stage, set the reference link to return multiple rows. In the database, consider setting a sparse lookup instead of a normal lookup. The Lookup stage retrieves records from the reference source. If you have a large number of input records, use a normal lookup. If you have a small number of input records (1:100 or more), use a sparse lookup. Typical real-time jobs have small numbers of input records. The Sparse lookup type queries the reference database for each input record that enters the job. The Normal lookup type transfers all the records from the reference database and then retrieves the records that are indicated by the input data. Because each of those transactions has a performance cost, the number of input records that are likely to enter the ISD Input stage determine which lookup type to use. If you have a small number of input records, the performance cost of querying for each record is less. When the sparse lookup queries for each input record, you save the amount of time required to transfer a large number of reference records to the Lookup stage. If you have a large number of input records, you pay the performance cost of a normal lookup that transfers all the reference records to the Lookup stage. However, that cost is less than the cost you pay if the Lookup stage queries the reference database for each input record.

- If you want to use a static view of your reference data as input to the match, use a fixed data source (such as a sequential file) and a Lookup stage. Use this job design if you want reference data that is a snapshot of a particular state of your database environment. As with the dynamic view of reference data, use a processing stage such as the Copy stage to copy data from the ISD Input stage to the Lookup and Two-source Match stages.

**Note:** In the case of either job design that does not use a Lookup stage to retrieve reference data, a stage that directly inputs reference data might exhibit unpredictable behavior in a real-time job. In such a design, the behavior is unpredictable because the ISD Input stage does not drive the flow of the reference data into the Two-source Match stage. Depending on the situation, a job designed in this way might return incorrect reference data or might fail to run. A common behavior in this job design is that the first request works but the second request returns nothing.

- Frequency data should reflect a representative sample of your larger data source. The frequency data generated during any one instance of a match job might not reflect a representative sample. Do not generate frequency data in real time. Because generating frequency data is time consuming, running it with every service request of a real-time job would reduce performance.

Related concepts:
- [Database sparse lookup vs. join](#)

Matching with a dynamic view of reference data

This method sets up a real-time Two-source Match job in which the input data records come from the ISD Input stage. The reference records are selected by a Lookup stage from a database, potentially allowing for a dynamic view of reference data.
About this task

The following figure shows a job design with a dynamic view of the reference records from the database where the reference records are maintained. The Lookup stage retrieves the reference records with each service request.

Note: The following example uses one Lookup stage to retrieve reference records from the database. If you use a match specification in the Two-source Match stage that contains more than one pass, two approaches are possible. The first approach would use a single lookup to select reference records required for all passes using a single query that considers the blocking columns for each pass. Alternately, multiple lookups which select reference records for each pass based on the blocking criteria per pass could be used with the resulting records aggregated and fed into the reference match. In the first case an explicit SQL query would need to be written that would potentially be more complex than those required in the second case.

Procedure

1. From the Data Quality palette, select the Two-source Match stage and drag it on the middle of the canvas.
2. From the Processing palette, select the Lookup stage and drag it on the canvas above the Two-source Match stage. Link it to the Two-source Match stage.
3. From the Processing palette, select the Copy stage and drag it on the canvas to the left of the Two-source Match stage. Link it to the Two-source Match stage and the Lookup stage.

4. From the Real Time palette, select the ISD Input stage and drag it on the canvas below the Copy stage. Link it to the Copy stage.

5. From the Database palette, select the ODBC Connector stage or other database stage and drag it on the canvas to the left of the Lookup stage. Link it to the Lookup stage. The link shows as a dotted line because it is a reference link.

6. From the Real Time palette, select the ISD Output stage and drag it on the canvas to the right of the Two-source Match stage. Link the Two-source Match stage to the ISD Output stage.

7. From the File palette, select the Data Set stage and drag it on the canvas below the Two-source Match stage. Link it to the Two-source Match stage. The Data Set stage is used to illustrate a complete example that shows one stage from which the frequency data can be input into the match.

8. Drag another Data Set stage on the canvas to the right of the other Data Set stage. Link it to the Two-source Match stage also.

9. Configure the ISD Input stage.

10. Configure the Copy stage. For more information about the Copy stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

11. Configure the Lookup stage. For more information about the Lookup stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

12. Configure the database stage. For more information about the ODBC Connector stage, see IBM InfoSphere DataStage and QualityStage Connectivity Guide for ODBC. For more information about Enterprise database stages, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

13. Configure the Data Set stages to input the frequency data to the Two-source Match stage. For more information about the Data Set stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

14. Configure the Two-source Match stage. For more information about the Two-source Match stage, see IBM InfoSphere QualityStage User’s Guide.

15. Configure the ISD Output stage.

Related concepts:
- Lookup Stage Conditions
- Matching two sources: Reference Match stage
- Lookup Stage
- Copy stage
- ODBC connector jobs

Related tasks:
- Changing the lookup operation type in the connector

Matching with a static view of reference data

This method sets up a real-time Two-source Match job in which the input data records come from the ISD Input stage. The reference records that you want to match are stored in a fixed data source that provides a snapshot of your reference data. The snapshot captures the contents of your reference source at a particular point in time.
About this task

The following figure shows a job design with a static view of the reference records.

![Diagram of a real-time Two-source Match job](image)

**Figure 6. A real-time Two-source Match job (formerly called Reference Match job) that selects static reference data**

**Procedure**

1. From the Data Quality palette, select the Two-source Match stage and drag it on the middle of the canvas.
2. From the Processing palette, select the Lookup stage and drag it on the canvas above the Two-source Match stage. Link it to the Two-source Match stage.
3. From the Processing palette, select the Copy stage and drag it on the canvas to the left of the Two-source Match stage. Link it to the Two-source Match stage and the Lookup stage.
4. From the Real Time palette, select the ISD Input stage and drag it on the canvas below the Copy stage. Link it to the Copy stage.
5. From the File palette, select the Sequential File stage or other file stage and drag it on the canvas to the left of the Lookup stage. Link it to the Lookup stage. The link shows as a dotted line because it is a reference link.
6. From the Real Time palette, select the ISD Output stage and drag it on the canvas to the right of the Two-source Match stage. Link the Two-source Match stage to the ISD Output stage.
7. From the File palette, select the Data Set stage and drag it on the canvas below the Two-source Match stage. Link it to the Two-source Match stage. The Data Set stage is used to illustrate a complete example that shows one stage from which the frequency data can be input into the match.

8. Drag another Data Set stage on the canvas to the right of the other Data Set stage. Link it to the Two-source Match stage also.

9. Configure the ISD Input stage.

10. Configure the Copy stage. For more information about the Copy stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

11. Configure the Lookup stage. For more information about the Lookup stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

12. Configure the Sequential File or other file stage. For more information about the file stages, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

13. Configure the Data Set stages to input the frequency data to the Two-source Match stage. For more information about the Data Set stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

14. Configure the Two-source Match stage. For more information about the Two-source Match stage, see IBM InfoSphere QualityStage User’s Guide.

   In addition to setting the parameters for the stage properties, select Allow Multiple Instances and Enabled for Information Services from Edit > Job Properties.

15. Configure the ISD Output stage.

Related concepts:

- Lookup Stage Conditions
- Matching two sources: Reference Match stage
- Copy stage
- Lookup Stage

Matching with pre-selected input and reference data

This method sets up a real-time Two-source Match job in which the input data records and input reference records come from the ISD Input stage. The data and reference records are selected before they enter the real-time job.

Before you begin

Select records that you want to feed into the ISD Input stage as data or reference records. For example, you might add a field to your combined input data in which you indicate which record is a data record and which record is a reference record. With that indication, a stage (such as the Filter stage) is able to split data records to one output link and reference records into another output link. Both output links feed into the Two-source Match stage.

About this task

The benefit of this method is that you feed the data records and reference records in a single service request into the job by using the ISD Input stage. The drawback of this method is that you must use some method to select records as data or reference before they enter the real-time job. This job design might be a good option if your data and reference records are already selected for another purpose in your environment. The following figure shows this method of setting up a real-time job.
**Procedure**

1. From the Data Quality palette, select the Two-source Match stage and drag it on the middle of the canvas.
2. From the Processing palette, select the Filter stage and drag it on the canvas above the Two-source Match stage. Create two links to the Two-source Match stage. One link is for the reference data and one link is for the input data. The Filter stage is one example of a stage that can split the input data and reference data.
3. From the Real Time palette, select the ISD Input stage and drag it on the canvas to the left of the Filter stage. Link it to the Filter stage.
4. From the Real Time palette, select the ISD Output stage and drag it on the canvas to the right of the Two-source Match stage. Link the Two-source Match stage to the ISD Output stage.
5. From the File palette, select the Data Set stage and drag it on the canvas below the Two-source Match stage. Link it to the Two-source Match stage.

*Figure 7. A real-time Two-source Match job (formerly known as a Reference Match job) for which data records and reference records are selected before entering the job*
6. Drag another Data Set stage on the canvas to the right of the other Data Set stage. Link it to the Two-source Match stage also. The Data Set stage is used to illustrate a complete example that shows one stage from which the frequency data can be input into the match.

7. Configure the ISD Input stage.

8. Configure the Filter stage (or whichever stage that you use instead of this one). For more information about the Filter stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide.

9. Configure the Data Set stages to input the frequency data to the Two-source Match stage. For more information about the Data Set stage, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide.

10. Configure the Two-source Match stage. For more information about the Two-source Match stage, see IBM InfoSphere QualityStage User's Guide.

   In addition to setting the parameters for the stage properties, select Allow Multiple Instances and Enabled for Information Services from Edit > Job Properties.

11. Configure the ISD Output stage.

Related concepts:

Matching two sources: Reference Match stage
Filter Stage

---

**Timeout value for ISD Input and Output stages**

The **Timeout** value is the maximum time that the ISD Input stage or the ISD Output stage waits to communicate with the IBM InfoSphere Information Services Director Agent.

The default value of 30 seconds is adequate for most environments.

The **Timeout** value for the ISD Input stage and ISD Output stage must accommodate the following startup costs:

- Job control logic
- Before-job subroutines
- Before-stage subroutines of the ISD Input stage and ISD Output stage

**Related tasks:**

"Adding the ISD Input stage” on page 15
You configure the ISD Input stage while building a job in IBM InfoSphere DataStage and QualityStage Designer.

"Adding the ISD Output stage” on page 15
You configure the ISD Output stage while building a job in IBM InfoSphere DataStage and QualityStage Designer.
Chapter 4. Connecting to information service providers

After you create an information service provider, you must connect to it before you can use it to develop applications, services, and operations.

About this task

There are four information providers types — a DataStage and QualityStage type for IBM InfoSphere DataStage and IBM InfoSphere QualityStage; a DB2 or Federation Server type for IBM DB2 or IBM InfoSphere Federation Server; a Classic Federation Server type for IBM InfoSphere Classic Federation Server for z/OS; and an Oracle Database Server type for Oracle Database Server.

Connecting to InfoSphere DataStage and QualityStage

Complete this task to connect to the following information service providers: IBM InfoSphere DataStage or IBM InfoSphere QualityStage.

Before you begin

Before you connect to InfoSphere DataStage or InfoSphere QualityStage, you must complete the following task:

- Create an information service provider

Procedure

1. From the IBM InfoSphere Information Server Console, select Home > Configuration > Information Services Connections.
2. Select an item from the list and click Open from the Tasks pane. This list will be pre-populated with all the InfoSphere DataStage and InfoSphere QualityStage servers registered to your domain.
3. Click Edit to switch to the edit mode on the right corner of the screen.
4. Enter the user name and password for the server you want to connect to.
5. Fill out the form with the following information:
   - Connection Name: Name given to this connection.
   - Information Provider Type: Select DataStage and QualityStage.
   - Agent Host: Name of server running the ASB Agent.
   - Database Host: Name of the server where the provider resides.
   - Port: Default for InfoSphere DataStage and InfoSphere QualityStage is 31538.
6. Click Save and Enable, and then click Close to return to the Information Services Connection window.

Connecting to IBM DB2, IBM InfoSphere Federation Server, IBM InfoSphere Classic Federation Server for z/OS, or Oracle Database Server

Complete this task to connect to the following information service providers: IBM DB2, IBM InfoSphere Federation Server, IBM InfoSphere Classic Federation Server for z/OS, or Oracle Database Server.
Before you begin

Before you connect to IBM DB2 or IBM InfoSphere Federation Server, you must complete the following task:

- Create an information service provider

About this task

If you use Oracle Database Server as an information provider, refer to the JDBC driver documentation for details on setting up JDBC connection properties. DataDirect Connect for JDBC: User’s Guide and Reference is available at www.datadirect.com.

Procedure

1. In the Information Services Connection window click New in the Tasks pane.
2. Fill out the form with the following information:
   - Connection Name: Name given to this connection.
   - Information Provider Type: Select Classic Federation Server, DB2 or Federation Server, or Oracle Database Server.
   - Agent Host: Name of server running the ASB Agent.
   - Database Host: Name of the server where the provider resides.
   - Port: Default for IBM DB2 or IBM InfoSphere Federation Server is 50000 unless you have multiple versions of IBM DB2 running. Default for DataStage or QualityStage is 31538.
3. Click Add to add databases to the list of databases. Provide database details along with the user name and password to access it, and click OK.
4. Fill out the database form with the following information:
   - Database
   - User Name
   - Password
5. Click OK.

Note: JDBC Connection Properties such as isolation levels may be specified to override defaults used by IBM InfoSphere Information Server.

6. Click Save and Enable, and then Close.
Chapter 5. Registering services from IBM InfoSphere MDM Server

If you want to register services that are available on IBM InfoSphere Master Data Management Server, register the server and import the interchange archive that is created by using IBM InfoSphere MDM Server Workbench. IBM InfoSphere MDM Server provides central management of master data across customer, product, and account domains.

Before you begin

To register a service from IBM InfoSphere MDM Server, you must be an IBM InfoSphere Information Services Director Administrator and you must complete the following task:

- Create an interchange archive that contains the service by using IBM InfoSphere MDM Server Workbench. Refer to the IBM InfoSphere Master Data Management Server Workbench User Guide for details on creating an interchange archive.

About this task

When you register services from IBM InfoSphere MDM Server into IBM InfoSphere Information Server, you can view them as deployed services in the Information Services Catalog tab of the IBM InfoSphere Information Server Web console.

Procedure

1. In the IBM InfoSphere Information Server console, click HOME > Configuration > Master Data Services Registration.
2. In the Tasks pane, click New.
3. In the Master Data Server Host field, enter the name of the computer that runs IBM InfoSphere MDM Server.
4. In the Port field, enter the port for the server.
5. In the MDM Services Interchange Archive field, click Browse.
6. Select the MDM Archive file and click Open.
7. Optional: Enter a name for the primary contact for the published services.
8. Optional: Enter the e-mail address of the primary contact for the published services.
9. Optional: Click Test to verify that you can access the host and services.
10. Click Save > Save and Close.
Chapter 6. Developing applications, services, and operations

As shown in the following figure, the IBM InfoSphere Information Services Director development model is based on a hierarchy of objects that define the flow of development.

![Development model diagram](image)

Figure 8. Development model

InfoSphere Information Services Director development requires the definition of:

**Projects**
A project is a collaborative environment that contains applications, services, and operations. You can define some access rights at the project level such as the users who can design or deploy applications.

**Applications**
An application is a container for a set of services. Because the application is the unit of deployment, all services within a single application must be deployed or undeployed together.

**Services**
A service is composed of one or more operations. Creating services is the point of the development process at which you attach bindings. Choose a binding based on the method you use to invoke the operations from your client. Services expose results from processing by information providers such as IBM InfoSphere DataStage servers and federated servers.

**Operations**
The business logic for your information service is contained in the operations. The operation describes the actual task performed by the information provider. Examples of operations include an InfoSphere QualityStage or InfoSphere DataStage job, an IBM InfoSphere Federation Server query or invoking a stored procedure in an IBM DB2 database.
Related concepts:
Chapter 1, “Introduction to IBM InfoSphere Information Services Director,” on page 1
You can more easily create business processes with information from a complex, heterogeneous environment if that information is published as consistent and reusable services.

Related tasks:
Chapter 2, “Supported information service providers,” on page 5
Information service providers are the sources of operations for your services. Using IBM InfoSphere Information Services Director, you can create services from the following information service providers: IBM InfoSphere DataStage; IBM InfoSphere QualityStage; IBM DB2 for Linux, UNIX, and Microsoft Windows; IBM InfoSphere Federation Server; IBM InfoSphere Classic Federation Server for z/OS; and Oracle Database Server.

Creating a project

A project is a collaborative environment that contains applications, services, and operations. You can define some access rights at the project level such as the users who can design or deploy applications.

Before you begin

Before you create a project, you must be logged in to the IBM InfoSphere Information Server console as an IBM InfoSphere Information Services Director administrator.

Procedure

1. Click New Project in the bottom right corner of the My Home workspace.
2. Select Information Services as the project type. If only InfoSphere Information Services Director is installed, ignore this step.
3. Type a name for the project.
4. Click OK.

Related tasks:
“Creating an application” on page 31
An application is a container for a set of services. Because the application is the unit of deployment, all services within a single application must be deployed or undeployed together.

“Creating a service” on page 33
Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

“Adding operations” on page 57
Add operations to define the information tasks of a service. Each information service is composed of a set of operations that use artifacts such as database queries for its business logic. The implementation of operations, such as database queries and IBM InfoSphere DataStage jobs, is hidden from the service client. This loose coupling enables service clients to be developed independently of information services, and requires minimal coordination between the service developer and the service client developer.

About projects

A project is a container for applications. It is a logical container for all of the tasks that can be performed using IBM InfoSphere Information Services Director.

Before services can be created and deployed, you need to create at least one project and one information services application. One project can hold many applications, and in each application, many services can be created.
All project information that you create by using InfoSphere Information Services Director is saved in the common metadata repository so that it can easily be shared among other IBM InfoSphere Information Server components.

Tip: You can export a project to back up your work or share work with other InfoSphere Information Server users. The export file includes applications, services, operations, and binding information.

Creating an application

An application is a container for a set of services. Because the application is the unit of deployment, all services within a single application must be deployed or undeployed together.

Before you begin

**“Creating a project” on page 30**

Procedure

1. Select a project from the Projects pane of the My Home workspace and click Open Project.
2. Select Develop > Information Services Director > Information Services Application to bring you to the Information Services Application workspace. If only IBM InfoSphere Information Services Director is installed, select Develop > Information Services Application.
3. Click New in the Tasks pane
4. In the application Overview pane, you must type a name for the application in the Name field. The length of the application name must be limited to 25 characters or less. All the other required fields contain default information. The description and e-mail address fields are optional.
5. Click Save Application.
6. Click Close Application.

Related tasks:

**“Creating a project” on page 30**
A project is a collaborative environment that contains applications, services, and operations. You can define some access rights at the project level such as the users who can design or deploy applications.

**“Creating a service” on page 33**
Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

**“Adding operations” on page 57**
Add operations to define the information tasks of a service. Each information service is composed of a set of operations that use artifacts such as database queries for its business logic. The implementation of operations, such as database queries and IBM InfoSphere DataStage jobs, is hidden from the service client. This loose coupling enables service clients to be developed independently of information services, and requires minimal coordination between the service developer and the service client developer.

About applications

An application contains one or more services that you want to deploy together as an enterprise archive (EAR) file on an application server.

To edit an existing application, you must first click Edit in the application Overview pane. While you are editing the application in the edit mode, the application is locked and can not be edited by another user at the same time.

All design-time activity occurs in the context of applications:

- Creating services and operations
• Describing how message payloads and transport protocols are used to expose a service
• Attaching a reference provider, such as an IBM InfoSphere DataStage job or a SQL query, to an operation

Importing RTI files

IBM InfoSphere Information Services Director also recognizes RTI export format files, also known as RTIX files.

About this task

This procedure shows how to migrate from IBM WebSphere RTI 7.5 and 7.5.1 to InfoSphere Information Services Director.

Procedure

1. Use the WebSphere RTI Export Wizard to create an RTIX file. This RTIX file contains descriptions of operations and services.
2. The RTIX file is then imported using the IBM InfoSphere Information Server console Import function. This imported file is the equivalent of the output of the IBM InfoSphere Information Server console design function.
3. Imported service descriptions must then be associated with an application object before they are deployed. The import function is done at the application level to create this association.

Results

At this point the imported service description is the equivalent of the service that has passed through the design process in IBM InfoSphere Information Server and is deployed in the same way as any natively designed information service.

Importing an application

You can import an application to accomplish tasks such as migrating applications from a development to a production environment.

Procedure

1. From the IBM InfoSphere Information Server console, select a project from the Projects pane and click Open Project.
2. Click DEVELOP > Information Services Application to open the Information Services Application workspace.
3. From the Tasks pane, click Import.
4. Select the file that you want to import and click Open.
5. In the Import Application window, click Import.

Results

You deploy an imported application in the same way as an application that you create by using IBM InfoSphere Information Services Director and the IBM InfoSphere Information Server console.

Exporting an application

During development, IBM InfoSphere Information Services Director provides the ability to export and import projects, applications, and specific services.
About this task

You can also export the services defined in an application before it is deployed and import the services into another application. When you are exporting your InfoSphere Information Services Director project or applications you have the choice of merely exporting only the design-time information, which creates an XML file, or you can export the application runtime information, which creates a DAT file.

Procedure

1. Select the application in the Information Services Application workspace.
2. In the Tasks pane, click Export. The runtime information includes all the metadata required by the application server, the metadata repository, and the enterprise archive (EAR) file.

Creating a service

Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

Before you begin

“Creating a project” on page 30
“Creating an application” on page 31

About this task

With IBM InfoSphere Information Services Director 8.1, you add basic HTTP authentication, user name token authentication, or HTTPS confidentiality to a service by selecting Requires Authentication, Requires Confidentiality, or both when you create a service within an application.

Procedure

1. Select a project from the Projects pane of the My Home workspace.
2. Go to the Information Services Application workspace by choosing one of the following options:

<table>
<thead>
<tr>
<th>Installed IBM InfoSphere Information Server component</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If only InfoSphere Information Services Director is installed</td>
<td>Select Develop &gt; Information Services Application.</td>
</tr>
<tr>
<td>If InfoSphere Information Services Director and IBM InfoSphere Information Analyzer are installed</td>
<td>Select Develop &gt; Information Services Director &gt; Information Services Application</td>
</tr>
</tbody>
</table>

3. Select an application from the Select Information Services Application to Work With pane and click Open.
4. Click Edit in the bottom right corner of the Overview pane.
5. At the bottom of the Select a View pane, click New > Service.
6. In the service Overview pane, you must type a name for the service in the Service Name field. All the other required fields contain default information. The description and e-mail address fields are optional.
7. Optional: If you want to add security to the service, select Requires Authentication, Requires Confidentiality, or both.
8. Click Save Application.
9. Click Close Application.
Related concepts:

“EJB service binding” on page 48
If your service consumers want to access an information service through an EJB interface, attach the EJB binding to the information service.

“Attaching a binding to a service” on page 35
Service consumers are able to access information services using multiple technologies for program interoperability. These technologies are called bindings.

“SOAP over HTTP service binding” on page 42
To expose an information service as a Web service, attach the SOAP over HTTP binding to the information service.

“REST service binding” on page 50
Use the REST binding to access an information service through a simple HTTP interface and generate a response in either XML or JSON format. IBM InfoSphere Information Services Director supports GET and POST methods.

“Request and response formats for the RSS binding” on page 50
Use the request formats to invoke a service with an RSS binding. Use the response formats to map the return arguments of the service operation to the channel and item attributes.

“About services”
An information service is a container for a group of related operations. When you create services, you indicate the method that you use to invoke operations within a service. In addition, you indicate your security requirements at the service level.

Related tasks:

“Creating a project” on page 30
A project is a collaborative environment that contains applications, services, and operations. You can define some access rights at the project level such as the users who can design or deploy applications.

“Creating an application” on page 31
An application is a container for a set of services. Because the application is the unit of deployment, all services within a single application must be deployed or undeployed together.

“Adding operations” on page 57
Add operations to define the information tasks of a service. Each information service is composed of a set of operations that use artifacts such as database queries for its business logic. The implementation of operations, such as database queries and IBM InfoSphere DataStage jobs, is hidden from the service client. This loose coupling enables service clients to be developed independently of information services, and requires minimal coordination between the service developer and the service client developer.

“Setting up JMS in IBM WebSphere Application Server” on page 35
If you attach the SOAP Over Java Message Service (JMS) binding or the Text Over JMS binding by using IBM InfoSphere Information Services Director in the IBM InfoSphere Information Server console, you must first configure messaging queues or topics within IBM WebSphere Application Server. Queues and topics are the virtual places where messages are exchanged.

About services
An information service is a container for a group of related operations. When you create services, you indicate the method that you use to invoke operations within a service. In addition, you indicate your security requirements at the service level.

An information service is a collection of operations that are selected from jobs, federated queries, or other information providers. You can group operations in the same information service or design them in separate services.
Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job topologies for different load and work style requirements: batch jobs, batch jobs with a service output stage, and jobs with service input and output stages.

Related tasks:
"Creating a service" on page 33
Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

Attaching a binding to a service

Service consumers are able to access information services using multiple technologies for program interoperability. These technologies are called bindings.

Bindings transfer messages between a service and a client. They are comprised of two parts: the format (SOAP, for example) and the transport method (HTTP, for example). The EJB binding defines both the format and the transport method.

IBM InfoSphere Information Services Director allows the same service to support multiple bindings. Bindings improve the utility of services and increase the likelihood of reuse and adoption across the enterprise.

The application that you develop to use services determines which binding you want to attach to that service. Many factors affect which binding you choose. For example, the EJB, SOAP Over JMS, and Text Over JMS bindings are easier to implement if your application is written in Java because EJB and JMS are Java specifications that do not have native implementations in C# or C++. The other bindings (REST, REST2, RSS, and SOAP Over HTTP) are common specifications that most languages can use easily.

Related concepts:
Chapter 1, “Introduction to IBM InfoSphere Information Services Director,” on page 1
You can more easily create business processes with information from a complex, heterogeneous environment if that information is published as consistent and reusable services.

Related tasks:
"Creating a service” on page 33
Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

Configuring a binding using JMS

Java Message Service (JMS) is an application programming interface that provides Java language functions for handling messages. Use JMS as the transport method when you want to send and receive those messages via message queues or topics. The format for the messages can be SOAP or you can use your own custom format with the text over JMS binding.

Setting up JMS in IBM WebSphere Application Server

If you attach the SOAP Over Java Message Service (JMS) binding or the Text Over JMS binding by using IBM InfoSphere Information Services Director in the IBM InfoSphere Information Server console, you must first configure messaging queues or topics within IBM WebSphere Application Server. Queues and topics are the virtual places where messages are exchanged.
About this task

These topics document a simple example of creating JMS request and response queues or topics. Refer to the IBM WebSphere Application Server, Version 6.1 information center or WebSphere Application Server, Version 7.0 information center for details on setting up more complex configurations.

Related tasks:
“Creating a service” on page 33

Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

Creating a service integration bus in IBM WebSphere Application Server:

The first step in setting up JMS in IBM WebSphere Application Server is to create the service integration bus, the virtual place where messages are exchanged. A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

About this task

Refer to the IBM WebSphere Application Server, Version 6.1 information center or WebSphere Application Server, Version 7.0 information center for details on creating a service integration bus.

Procedure

1. Log in to the IBM WebSphere Application Server administrative console.
2. Expand Service integration and click Buses.
3. Click New.
4. Enter a name and optional description for the service integration bus. For example, Bus1.
5. Clear the Secure check box but keep all the other default settings.
6. Click Apply.
7. Click the Save link at the top of the screen.
8. Click Save.

Adding IBM InfoSphere Information Server as a member of the service integration bus:

After you create a service integration bus, you must add IBM InfoSphere Information Server as a member. Bus members are the servers and clusters that have been added to the bus.

Before you begin

You must complete the following tasks:
• Create a service integration bus in IBM WebSphere Application Server

Procedure

1. In the IBM WebSphere Application Server administrative console, expand Service integration and click Buses.
2. In the Buses pane, click the name of the bus to which you will add InfoSphere Information Server as a member. For example, Bus1.
3. In the Topology panel, click Bus members.
4. Click Add.
5. Click Server and select the instance of IBM WebSphere Application Server on which InfoSphere Information Server runs. For example Node01:server1.
6. Click Next and Finish.
7. Click the Save link at the top of the screen.
8. Click Save.

Defining queues and topics in the service integration bus:

Queues and topic spaces are the two types of destinations within a bus. After you add IBM InfoSphere Information Server as a member of the bus, choose queue for point-to-point messaging, or topic space for publish and subscribe messaging. A bus destination is a virtual place within a service integration bus, to which applications attach as producers, consumers, or both to exchange messages.

Before you begin

You must complete the following tasks:

- Create a service integration bus in IBM WebSphere Application Server
- Add InfoSphere Information Server as a member of the bus

About this task

Refer to the IBM WebSphere Application Server, Version 6.1 information center or WebSphere Application Server, Version 7.0 information center for details on defining queues and topics in the service integration bus.

Procedure

1. In the IBM WebSphere Application Server administrative console, expand Service integration and click Buses.
2. In the Buses pane, click the name of the bus for which you will define queues or topics. For example, Bus1.
3. In the Destination Resources panel, click Destinations.
4. Click New.
5. Click Queue or Topic space and click Next.
6. Enter an identifier and an optional description for the queue or topic space. For example, RequestQueue.
7. Click Next until you reach the confirm window. Click Finish.
8. Repeat steps 4 through 7 until you have created the number of queues or topic spaces that you need for your application. At least two queues or topic spaces are recommended so that you have one for service requests and one for service responses.
9. Click the Save link at the top of the screen.
10. Click Save.

Associating JMS queues and topics with the service integration bus queues and topic spaces:

After you define the type of destination for your messages in the service bus, you must associate those queues or topic spaces with Java Message Service (JMS) queues or topics. A JMS queue is used as a destination for point-to-point messaging. A JMS topic is used for JMS publish and subscribe messaging. Use JMS destination administrative objects to manage JMS queues or topics for the default messaging provider.
Before you begin

You must complete the following tasks:

- Create a service integration bus in IBM WebSphere Application Server
- Add IBM InfoSphere Information Server as a member of the service integration bus
- Define request and response queues in the service integration bus

About this task

This task assumes that you are creating a JMS queue as the default messaging destination. If you want to use a JMS topic as the default messaging destination, click JMS topic instead of JMS queue in the Destinations panel of the Default messaging provider pane.

Refer to the IBM WebSphere Application Server, Version 6.1 information center or WebSphere Application Server, Version 7.0 information center for details on JMS queues and topics.

Procedure

1. In the IBM WebSphere Application Server administrative console, expand Resources > JMS Providers and click Default messaging.
2. In the Destinations panel, click JMS queue.
3. In the Default Messaging Provider pane, click New.
4. Enter a name for the JMS request queue. For example, MyRequestQueue.
5. Enter a JNDI name for the JMS request queue. As a convention, use a JNDI name of the form jms/Name, where Name is the logical name of the resource. For example, jms/MyRequestQueue.
6. In the Connection panel, select the bus name of the service integration bus.
7. Select the queue name of the service integration bus request queue.
8. Keep all the other default settings and click OK.
9. Repeat steps 3 through 8 until you have created the number of queues that you need for your application. At least two queues are recommended so that you have one for service requests and one for service responses.
10. Click the Save link at the top of the screen.
11. Click Save.

Creating a JMS connection factory:

After you associate the bus queues or topic spaces to the JMS queues or topics, you must create a Java Message Service (JMS) connection factory. A JMS connection factory is used to create connections to the associated JMS provider of JMS queues for point-to-point messaging, or JMS topics for publish and subscribe messaging. Use connection factory administrative objects to manage JMS connection factories for the default messaging provider.

Before you begin

You must complete the following tasks:

- Create a service integration bus in IBM WebSphere Application Server
- Add IBM InfoSphere Information Server as a member of the bus
- Define request and response queues in the bus
- Associate JMS queues with the service integration queues
### About this task

There are three types of JMS connection factories: a unified JMS connection factory, or the domain-specific JMS queue connection factory and JMS topic connection factory. Because the current JMS spec makes no distinction between JMS queue and JMS topic connection factory, this task documents the use of the unified JMS connection factory. If, however, your application is developed to use either a queue-specific or topic-specific interface, use the JMS connection factory that matches your application. For example, if you create a JMS connection factory and receive the following `ClassCastException`, try creating a JMS queue connection factory instead.

```
WSWS3406E: Unexpected exception caught while sending reply message:
java.lang.ClassCastException: com/ibm/ws/sib/api.jms/iml/JmsManaged
ConnectionFactoryImpl incompatible with javax/jms/QueueConnectionFactory
```

Refer to the [IBM WebSphere Application Server, Version 6.1 information center](https://www.ibm.com/support/pages/node/802860) or [WebSphere Application Server, Version 7.0 information center](https://www.ibm.com/support/pages/node/802864) for details on JMS connection factories.

### Procedure

1. In the IBM WebSphere Application Server administrative console, expand **Resources > JMS Providers** and click **Default messaging**.
2. In the Connection Factories panel, click **JMS connection factory**.
3. In the Default Messaging Provider pane, click **New**.
4. Enter a name for the JMS connection factory. For example, **MyQueueConnectionFactory**.
5. Enter a JNDI name for the JMS connection factory. For example, **jms/MyQueueConnectionFactory**.
6. In the Connection panel, select the bus name of the service integration bus.
7. Keep all the other default settings and click **OK**.
8. Click the **Save** link at the top of the screen.
9. Click **Save**.

### Creating a JMS activation specification:

After you create the Java Message Service (JMS) connection factory, you must create the JMS activation specification. A JMS activation specification is associated with one or more message-driven beans (MDB) and provides the configuration necessary for them to receive messages.

### Before you begin

You must complete the following tasks:

- Create a service integration bus in IBM WebSphere Application Server
- Add IBM InfoSphere Information Server as a member of the bus
- Define request and response queues in the bus
- Associate JMS queues with the service integration queues
- Create a JMS queue connection factory

### About this task

Refer to the [IBM WebSphere Application Server, Version 6.1 information center](https://www.ibm.com/support/pages/node/802860) or [WebSphere Application Server, Version 7.0 information center](https://www.ibm.com/support/pages/node/802864) for details on JMS activation specifications.

### Procedure

1. In the IBM WebSphere Application Server administrative console, expand **Resources > JMS Providers** and click **Default messaging**.
2. In the Activation Specifications panel, click **JMS activation specification**.
3. In the Default Messaging Provider pane, click **New**.

4. In the Administration panel, enter a name for the JMS activation specification. For example, *MyActivationSpecification*.

5. Enter a JNDI name for the JMS activation specification. For example, *jms/MyActivationSpecification*.

6. In the Destination panel, select **Queue** if the destination name in the next step is a queue. Select **Topic** if it is a topic.

7. In the **Destination JNDI name** field, enter the JNDI name of the JMS request queue or topic. For example, *jms/MyRequestQueue*.

8. In the **Bus name** menu, select the service integration bus. For example *Bus1*.

9. Keep all the other default settings and click **OK**.

10. Click the **Save** link at the top of the screen.

11. Click **Save**.

### Setting up JMS in IBM InfoSphere Information Services Director

If you want to send and receive messages for your service using the Java Message System (JMS) transport method, you must attach either the SOAP over JMS binding or the Text over JMS binding in the IBM InfoSphere Information Server console using IBM InfoSphere Information Services Director.

### Attaching the SOAP Over JMS binding:

Attach this binding to send and receive messages for your service in a SOAP format and transport them using Java Message Service (JMS).

**Before you begin**

- "Creating a project" on page 30
- "Creating an application" on page 31
- "Creating a service" on page 33

**Procedure**

1. From the Information Services Application workspace, open an application and click **Edit**.

2. From the Select a View pane of your application, select the service to which you want to attach the binding.

3. Double-click **Bindings**.

4. Click **Attach Bindings** at the bottom right corner of the screen and select **SOAP Over JMS**.

5. In the **Activation Spec JNDI Name** field, enter the JNDI name of the JMS activation specification that was specified when IBM WebSphere Application Server was set up for JMS. You can leave this field blank if you enter a name in the **Destination** field.

6. In the **Destination** field, enter the JNDI name of the JMS destination queue that was specified when IBM WebSphere Application Server was set up for JMS. You can leave this field blank if you enter a name in the **Activation Spec JNDI Name** field. If you enter values for both the activation spec JNDI name field and the destination field, the destination name will override the activation spec JNDI name. All other fields are optional or contain default values. Refer to the SOAP Over JMS binding properties topic for more information.

7. Click **Save Application**.

**SOAP Over JMS binding properties:**

This binding transports messages between a service and a client using the SOAP format over Java Message System (JMS) transport method.
As defined by the SOAP specification, enter the name of the response queue or topic in the replyTo JMS header of the request JMS message.

Table 2. General properties of the SOAP Over JMS binding

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP style</td>
<td>The encoding style for the SOAP messages.</td>
</tr>
<tr>
<td>SOAP action</td>
<td>The value included in the SOAPAction header of the SOAP message. Select NONE to use an empty header. Select OPERATION to include the name of the operation in the header.</td>
</tr>
</tbody>
</table>

Table 3. SOAP Over JMS binding properties that apply to the incoming (request) messages

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation Spec JNDI Name</td>
<td>The JNDI name of the JMS activation specification. If you enter a name in this field, you do not need to enter a name in the Destination field.</td>
</tr>
<tr>
<td>Destination</td>
<td>The JNDI name of the JMS destination queue. If you enter a name in this field, you do not need to enter a name in the Activation Spec JNDI Name field. You can leave this field blank if you enter an activation spec name. If you enter values for both the Activation Spec JNDI Name field and the Destination field, the destination name will override the activation spec JNDI name.</td>
</tr>
</tbody>
</table>

Table 4. SOAP Over JMS binding properties that apply to the outgoing (response) messages

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMS Destination Mode</td>
<td>Select Queue if you want only one reader to receive messages that are added to a queue. Select Topic if you want multiple readers who are subscribed to a topic to receive all the messages that are added to the topic. Ensure that the destination provided in the replyTo JMS header of the request message is a queue if you select Queue or a topic if you select Topic.</td>
</tr>
<tr>
<td>Time to Live</td>
<td>Select the number of milliseconds that a message will exist after it is sent. A value of 0 means that the message never expires. Refer to the IBM WebSphere Application Server, Version 6.0 Information Center for more information about the TimeToLive property reference.</td>
</tr>
<tr>
<td>Priority</td>
<td>Select a value between 0 and 9 that defines the priority of a message. Select a value between 0 and 4 for normal priority, and select a value between 5 and 9 for higher priority.</td>
</tr>
<tr>
<td>Connection Factory</td>
<td>Type the JNDI name of the connection factory that is used to send a reply message.</td>
</tr>
<tr>
<td>Message Selector</td>
<td>Enter a JMS message selector to filter the messages that are posted to the destination queue. Refer to the JMS specification for more information about the JMS message selector. Also, refer to the IBM WebSphere Application Server, Version 6.0 Information Center for information on message selection and filtering.</td>
</tr>
</tbody>
</table>

Attaching the Text Over JMS binding:
Attach this binding to send and receive messages in a format of your choice and transport them using Java Message Service (JMS).

**Before you begin**

“Creating a project” on page 30

“Creating an application” on page 31

“Creating a service” on page 33

**Procedure**

1. From the Information Services Application workspace, open an application and click **Edit**.
2. From the Select a View pane of your application, select the service to which you want to attach the binding.
3. Double-click **Bindings**.
4. Click **Attach Bindings** at the bottom right corner of the screen and select **Text Over JMS**.
5. Optional: Enter a description for the binding.
6. Click **Save Application**.

**SOAP over HTTP service binding**

To expose an information service as a Web service, attach the SOAP over HTTP binding to the information service.

If you attach the SOAP over HTTP binding to your service, you do not need to select any additional settings when you add an operation to the service unless you selected the **Requires Authentication** check box in the security Overview pane.

The SOAP Over HTTP binding pane contains the following optional settings:

- SOAP style and a SOAP action for this binding.
- Description of the binding.
- **Include this binding** check box. (To exclude the binding from the deployment of the application that contains this information service, clear this setting.)

The SOAP Over HTTP binding pane contains the following required setting (if you selected the **Requires Authentication** check box in the security Overview pane):

- Select a method of authentication support (HTTP basic or WS-Security username token).

**Related tasks:**

“Creating a service” on page 33

Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

**Text over HTTP service binding**

To publish an information service that flexibly accepts messages through HTTP, attach the Text over HTTP binding to the information service.
Overview of this binding

HTTP request and response payloads are passed “as is” to the underlying InfoSphere Information Services Director providers. To remain flexible, the Text over HTTP implementation makes no assumptions about the format of the HTTP request or response payload, so you can customize HTTP requests and payloads to use SOAP, XML, text, and so on.

Note: The underlying IBM InfoSphere Information Services Director provider (such as the IBM InfoSphere DataStage job) is responsible for understanding and composing this format.

The Text over HTTP binding supports many features:
- Ability to transform the request payload through XSLT style sheets that you provide, before it is sent to the information service providers and to transform the information service provider results by using XSLT before the results are returned in the HTTP response. XSLT 1.0 is the supported version.
- You can map transport data units (HTTP headers, HTTP request parameters, HTTP payload) of the HTTP protocol to and from InfoSphere Information Services Director service input and output arguments.
- Multi-part HTTP request support.
- You can receive and send complex SOAP messages with arbitrarily complex XML types. This ability can be required if the InfoSphere Information Services Director service must implement a predefined web service interface, for example, to implement a given WSDL.

The Text over HTTP binding pane includes options for the following features:
- Description of the binding.
- **Service Description File** path. Specify a description document, which can be used to provide documentation for service users. The documentation that you provide can explain how service users can call the service. This document can be in any format, such as HTML, XML, text, PDF (Portable Document Format) or .zip file. This document is packaged as part of the service and is accessible from the InfoSphere Information Services Director service catalog through a URL that service users can click. If you specify a WSDL document as the service description file, make sure to select the **WSDL Service Description** check box.
- **WSDL Service Description**. If you are providing a WSDL document as the service description document, select this option to flag the service description document as a WSDL document. This setting enables WSRR publishing capabilities at the InfoSphere Information Services Director service catalog level.

**Important**: You must make sure that the InfoSphere Information Services Director service can properly handle the SOAP requests that will be generated from the provided WSDL.

- **Include this binding** check box. To exclude the binding from the deployment of the application that contains this information service, clear this setting.
- Security options
  - **Requires Authentication** check box. If you select this option, the authentication mechanism uses HTTP Basic Authentication only. In particular, if the you use SOAP for the HTTP payload format, the SOAP specific authentication protocol (like WS-Security) is not supported because the protocol will require a true SOAP service to be generated in the application server.

  To force the authentication, all service operations require the InfoSphere Information Services Director Consumer role to be invoked. This role is declared at the service implementation level (all bean operations in the ejb-jar.xml of the session bean) and at the Text over HTTP servlet level (all URLs in the web.xml of the generated servlet). The URL to access the service description document is also protected by the InfoSphere Information Services Director Consumer role.

  To simplify administration in the InfoSphere Information Services Director service catalog, access to the service description document is also granted to the InfoSphere Information Services Director Administrator and InfoSphere Information Services Director Catalog Manager roles.
- **Requires Confidentiality** check box. If you select this option, a note will display that HTTPS/SSL is required to call this service. Only HTTPS is supported for confidentiality support. Specifically, if you use SOAP for the HTTP payload format, SOAP specific confidentiality protocol (WS-Security/XML encryption) is not directly supported by the Text over HTTP binding. You can still decide to pass WS-Security encrypted data directly down to the information provider if the information provider has the ability to decode it. However, the Text over HTTP binding does not support any automatic encryption or decryption of this data.

**Input mapping**

**Input Mapping.** You map HTTP protocol data units to service-input arguments. This process is HTTP request mapping.

InfoSphere Information Services Director supports only the following service operation input arguments:

- One or more scalar input arguments, for example, int, string, or boolean.
- A single complex type input argument that contains only scalar attributes.
- A single array of a complex type that contains only scalar attributes.

Define the mapping between the service input arguments and the HTTP protocol data units (HTTP headers, HTTP request parameters, or HTTP payload) in the following ways:

- Individual scalar input arguments can be mapped to any of the HTTP protocol data units.
- Individual scalar attributes of a complex type input argument can be mapped to any of the HTTP protocol data units.
- Individual scalar attributes of an array of complex type input arguments can be mapped to any of the HTTP protocol data units. In this case, an array with a single element is created for a given HTTP request and passed as the service input argument.

- **Type.** In the Text over HTTP mapping process, the HTTP request is mapped to service input arguments. The following HTTP protocol data units can be used for an HTTP request. For each scalar operation input argument, you must first select the type of HTTP protocol data unit (header, body, query parameters, and so on) that the value of the argument comes from. Then you can specify the HTTP data units that you want to use for the selected type:

**Fixed** You can provide a fixed value for any scalar operation service input argument. Regardless of the content of the incoming HTTP request, this specific input argument always has the entered value when the InfoSphere Information Services Director service is called.

**HTTP Payload**

If **HTTP Payload** is selected, the complete payload (the HTTP body) is passed as the service input argument.

Optionally provide an XSLT stylesheet to transform a valid XML payload before it is passed down to the mapped scalar operation input argument.

**HTTP Header**

Select a standard HTTP header or type a custom HTTP header name.

Some HTTP headers are available only for requests and responses. You must select the appropriate HTTP headers according to the HTTP protocol specification.

**HTTP Part**

Individual parts of the HTTP payload in a multipart/form-data request. Specify the name of the part that needs to match the name attribute of the Content-Disposition header of the part.

Optionally provide an XSLT stylesheet to transform a valid XML payload part before it is passed down to the mapped scalar operation input argument.
**HTTP Parameter**

Specify the name of a HTTP URL parameter. The HTTP parameter is encoded as part of the URL and is only supported for the HTTP GET method. (The POST method requires use of HTTP Payload.)

**CGI Variable and/or Servlet Variable**

Select a standard CGI variable and/or servlet variable. See Table 1 for a list of CGI variables and/or servlet variable properties that you can access as part of the HTTP request mapping.

*Table 5. CGI variable and/or servlet variable properties*

<table>
<thead>
<tr>
<th>HTTP Component</th>
<th>Mapping</th>
<th>HttpServletRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER_NAME</td>
<td>The server's hostname, DNS alias, or IP address as it would appear in self-referencing URLs.</td>
<td>getServerName()</td>
</tr>
<tr>
<td>SERVER_PROTOCOL</td>
<td>The name and revision of the information protocol this request came in with. Format: protocol/revision.</td>
<td>getProtocol()</td>
</tr>
<tr>
<td>SERVER_PORT</td>
<td>The port number to which the request was sent.</td>
<td>getServerPort()</td>
</tr>
<tr>
<td>Scheme</td>
<td>Returns the name of the scheme used to make this request, for example, HTTP or HTTPS.</td>
<td>getScheme()</td>
</tr>
<tr>
<td>REQUEST_METHOD</td>
<td>The method with which the request was made. For HTTP, this is “GET”, “HEAD”, “POST”, and so on.</td>
<td>getMethod()</td>
</tr>
<tr>
<td>PATH_INFO</td>
<td>The extra path information, as given by the client. In other words, scripts can be accessed by their virtual pathname, followed by extra information at the end of this path. The extra information is sent as PATH_INFO. This information should be decoded by the server if it comes from a URL before it is passed to the CGI script.</td>
<td>getPathInfo()</td>
</tr>
<tr>
<td>PATH_TRANSLATED</td>
<td>The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.</td>
<td>getPathTranslated()</td>
</tr>
<tr>
<td>Servlet Path</td>
<td>Returns the part of this request's URL that calls the servlet. This is the same as the SCRIPT_NAME CGI variable.</td>
<td>getServletPath()</td>
</tr>
<tr>
<td>QUERY_STRING</td>
<td>The query information is the information that follows the question mark (?) in the URL that referenced this script. The query information should not be decoded. This variable should always be set when there is query information, regardless of command line decoding.</td>
<td>getQueryString()</td>
</tr>
<tr>
<td>REMOTE_HOST</td>
<td>The host name making the request. If the server does not have this information, it should set REMOTE_ADDR and leave this unset.</td>
<td>getRemoteHost()</td>
</tr>
</tbody>
</table>
### Table 5. CGI variable and/or servlet variable properties (continued)

<table>
<thead>
<tr>
<th>HTTP Component</th>
<th>Mapping</th>
<th>HttpServletRequest</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_ADDR</td>
<td>The IP address of the remote host making the request.</td>
<td>getRemoteAddr()</td>
</tr>
<tr>
<td>AUTH_TYPE</td>
<td>If the server supports user authentication, and the script is protected, this is the protocol-specific authentication method used to validate the user.</td>
<td>getAuthType()</td>
</tr>
<tr>
<td>REMOTE_USER</td>
<td>If the server supports user authentication, and the script is protected, they have authenticated as this user name.</td>
<td>getRemoteUser()</td>
</tr>
<tr>
<td>Request URI</td>
<td>Returns the part of this request’s URL from the protocol name up to the query string in the first line of the HTTP request.</td>
<td>getRequestURI()</td>
</tr>
<tr>
<td>Request URL</td>
<td>Reconstructs the URL that the client used to make the request.</td>
<td>getRequestURL()</td>
</tr>
<tr>
<td>Context Path</td>
<td>Returns the portion of the request URI that indicates the context of the request (this is a servlet property and not a CGI variable).</td>
<td>getContextPath()</td>
</tr>
</tbody>
</table>

- The same HTTP request data units can also be mapped to different service input arguments. For example, the HTTP payload can be mapped to one or more different InfoSphere Information Services Director service input arguments, with a different XSLT stylesheet for each. This capability gives you the option to extract different parts of the HTTP payload using XSLT to provide specific values for different service input arguments.

### Output mapping

**Output Mapping.** You map service output arguments to the HTTP response. This process is HTTP response mapping.

InfoSphere Information Services Director supports only the following service operation output arguments:
- A single scalar output return argument, such as an int, string, or boolean return.
- A single complex type output argument that contains only scalar attributes.
- A single array of a complex type that contains only scalar attributes.

Use the mapping process to define the mapping between the service output argument and the HTTP response protocol data unit:
- Individual scalar output arguments, including complex types, can be mapped to any of the HTTP protocol data units.
- In a complex type output argument, individual scalar attributes of this complex type can be mapped to any of the HTTP protocol data units.
- In an “array of complex type” output argument, individual scalar attributes of this complex type can be mapped to any of the HTTP response protocol data units. Only the first element in the array is used to generate the HTTP response; any other elements in the array are ignored.
- **Type.** In the Text over HTTP mapping process, output arguments are mapped to HTTP response data units. The HTTP protocol data units supported for an HTTP response are as follows:
HTTP Header
Select a standard HTTP header or type a custom HTTP header name. Your options include the Content-Type header, which enables you to specify the media type of the HTTP response. If the Content-Type header is not mapped to any values, the application or octet-stream is used by default.

HTTP Payload
If selected, the value of the selected service output argument is taken "as is" and is used to set the content of the HTTP response payload (also called the HTTP response body).

Optionally provide an XSLT stylesheet to transform the service output argument before it is used to set the content of the HTTP response payload/body. The selected service output argument must be a valid XML document because XSLT will only understand an XML document as input.

Note: The same service output argument can also be mapped to different HTTP response data units. For example, the same scalar value of the service output argument can be used to set the value of an HTTP response header as well as of the HTTP response payload.

- XSLT. For any scalar value of the service output argument of type string, you can provide an XSLT stylesheet to transform the string before it is used to set the value of the mapped HTTP response data units. In this case, ensure that the underlying service provider returns the string as a valid XML document so that it can be properly handled by the XSLT style sheet.

- The combination of (1) mapping the same service output argument to different HTTP response data units and (2) using the XSLT support for string output values allows you to define a service operation that (a) can return a single string that can be a complex XML document and (b) extract different pieces of this XML document to assign values to the different HTTP response data units.

- You can provide the value of the HTTP content-type header that specifies the media-type of the HTTP payload. The HTTP payload and media-type must match. It is your responsibility to make sure that the payload and media-type match. Verify that the underlying information provider returns the data in the proper format, or use an XSLT stylesheet to transform the XML provider data to match the specified format.

Calling InfoSphere Information Services Director services by using the Text over HTTP binding

The URL syntax
- A unique base URL is used to identify which service an HTTP request is directed to. This base URL is defined as follows:
  http://servername:port/wisd-txhttp/<application-name>/<service-name> The path elements enclosed in < and > are replaced by the actual InfoSphere Information Services Director application name and service name.

- Some software and hardware might not allow certain characters in the URL.

- The name of the actual operation to call can be provided using either of these two options:
  - Adding the name of the operation at the end of the base URL:
    http://servername:port/wisd-txhttp/<application-name>/<service-name>/<operation-name>
  - Using the SOAPAction HTTP header to specify the name of the operation (in case the end-user decides to use text over HTTP with SOAP messages)

Both options are supported at all times. If both options are specified simultaneously, the first option (adding the operation name to the end of the base URL) takes precedence.

Error handling

A service calling error results in an HTTP error that is sent back to the client by the following rules:
HTTP 500 error (Internal Server Error)
Any unexpected errors that occur during the service invocation results in an HTTP 500 error (Internal Server Error) with the stack trace of the error provided as the response HTTP payload. This includes any XSLT stylesheet processing errors.

There is no automated support for SOAP fault generation in the event that SOAP/HTTP is used as the HTTP payload format with this binding. Using XSLT, you can examine the data that is returned by the underlying provider and possibly generate an HTTP response containing a SOAP fault. In the event of a Java exception during the processing of a request, no SOAP fault will be generated; only an HTTP error will be sent back to the caller.

HTTP 404 error (Not Found)
An attempt to access an operation that does not exist will result in an HTTP 404 error (Not Found).

HTTP 400 error (Bad Request)
Sending invalid data will result in an HTTP 400 error (Bad Request). For example, if the wrong service input arguments are used and cannot be properly decoded, such as sending a string while an integer is expected, the client receives this error.

Unsupported combinations of HTTP methods and content type will also result in an HTTP 400 error.

HTTP 405 error (Method Not Allowed)
Using an HTTP GET operation when only a POST is allowed will result in an HTTP 405 error (Method Not Allowed).

EJB service binding
If your service consumers want to access an information service through an EJB interface, attach the EJB binding to the information service.

If you attach the EJB binding to your service, you do not need to select any additional settings when you add an operation to the service. The EJB binding contains the following settings:

- Java Naming and Directory Interface (JNDI) Name – the JNDI name to be used to locate the EJB that is deployed on the application server for this binding.
- Package Name – the Java package name to be used when generating the Java client classes for the EJB binding.
- Include this binding check box. (To exclude the binding from the deployment of the application that contains this information service, clear this setting.)
- Security – If authentication is required, HTTP basic authentication is used. If confidentiality is required, HTTPS is used and must be configured in the application server.

Related tasks:
"Creating a service" on page 33
Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

Setting up RSS in IBM InfoSphere Information Services Director
To expose an operation as an RSS feed, you must attach the RSS binding in the IBM InfoSphere Information Server console using IBM InfoSphere Information Services Director.

Attaching the RSS binding
Attach the RSS binding to expose IBM InfoSphere Information Services Director services as RSS 2.0 feeds.
Before you begin

“Creating a project” on page 30
“Creating an application” on page 31
“Creating a service” on page 33

About this task

Restrictions

You can attach the RSS binding only to services that use scalar input arguments or no input arguments for all of their operations. RSS feeds are retrieved using HTTP GET URLs which means that input data can only be encoded as part of the URL parameters. Only scalar input arguments are supported because they are the only ones that can be encoded as a part of the URL. If the service requires some scalar input arguments, for each input argument the URL must contain a parameter with a name that matches the service argument name.

If you selected Requires Authentication when you created the service, you must secure the service with HTTP basic authentication. To invoke this secured service, you must have the role of Information Services Director Consumer.

Procedure

1. From the Information Services Application workspace, open an application and click Edit.
2. From the Select a View pane of your application, select the service to which you want to attach the binding.
3. Double-click Bindings.
4. Click Attach Bindings at the bottom right corner of the screen and select RSS.
5. Optional: Enter a description of the binding.
6. Clear the Include this binding check box to exclude the binding from the deployment of the application that contains this service.
7. Click Save Application.

Configuring settings for the RSS binding

If you attached the RSS binding to a service, you can add channel attributes and item attributes to the RSS feed.

Before you begin

You must complete the following tasks:

• “Creating a project” on page 30
• “Creating an application” on page 31
• “Creating a service” on page 33
• “Attaching the RSS binding” on page 48
• “Adding operations” on page 57

Procedure

1. From the Information Services Application workspace, open an application and click Edit.
2. From the Select a View pane of your application, expand Services > your service > Operations.
3. Double-click each operation of the service to configure the RSS binding properties.
4. Click the RSS Settings tab.
5. Click the **Channel Details** tab to select attributes. **TITLE**, **LINK**, and **DESCRIPTION** are required channel attributes.

6. Click the **Item Details** tab to select attributes. **TITLE** is a required attribute if you did not provide a description when you attached the RSS binding to the service. **DESCRIPTION** is a required attribute if you did not provide a **TITLE** attribute.

7. Click **Save Application** to save changes.

8. Click **Close Application** to close the application.

**Request and response formats for the RSS binding**

Use the request formats to invoke a service with an RSS binding. Use the response formats to map the return arguments of the service operation to the channel and item attributes.

**Related tasks:**

“Creating a service” on page 33

Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

**Request formats:** Use one of the following URLs to invoke an RSS service:

**Base URL formation to invoke an RSS service**

`http://servername:port/wisd-rss/applicationName/serviceName/operationName`

**URL formation for service operations with input arguments**

For this example, note that the names of the URL parameters (`argument1` and `argument2`) must match the names of the service input arguments. `http://servername:port/wisd-rss/applicationName/serviceName/operationName?argument1=value1&argument2=value2`

**Response formats:** The RSS 2.0 XML document is composed of:

- One channel element with multiple sub-elements for the channel properties
- One or more item elements that each represent one entry of the channel. An item element can have multiple sub-elements for the item properties.

**REST service binding**

Use the REST binding to access an information service through a simple HTTP interface and generate a response in either XML or JSON format. IBM InfoSphere Information Services Director supports GET and POST methods.

The REST binding contains the following settings:

- **Description** – Description of the binding.
- **Style Sheet File** – You can include a single XSLT style sheet when you attach the binding. If you include a style sheet file, a processing instruction referencing this style sheet is added to the responses in XML format, which allows consumers of the service (for example, a Web browser) to automatically apply the style sheet on the response. The style sheet is not used for responses in JSON format.
- **Include Style Sheet** check box – Click to include an optional XSLT style sheet.
- **Include this binding** check box. (To exclude the binding from the deployment of the application that contains this information service, clear this setting.)
- **Security** – If authentication is required, HTTP basic authentication is used. If confidentiality is required, HTTPS is used and must be configured in the application server.
Related tasks: "Creating a service" on page 33

Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

REST 2.0 service binding

Use the REST 2.0 binding to access an information service through a simple HTTP interface and generate a response in either XML or JSON format. IBM InfoSphere Information Services Director supports GET, POST, PUT, and DELETE actions for the REST 2.0 binding.

When you attach a REST 2.0 binding, settings must be configured at the services level and operation level.

The REST 2.0 binding contains the following settings:

- **Description** – Description of the binding.
- **Response Code Header** option. Use this option to specify that service invocation errors must be returned as HTTP 200 responses.
  
  If this property is set, any exception thrown by the service invocation is returned as an HTTP 200 response. The custom HTTP header, `X-HTTP-Status-Override`, will be set in the HTTP response to the actual HTTP status code (HTTP 500, by default).
- **Include this binding** option. To exclude the binding from the deployment of the application that contains this information service, clear this setting.

- **Operation Bindings Default Settings**
  
  You set the default values to be used in operation bindings in the **Operation Bindings Default Settings** area. This provides you with the ability to set the value once at the service level for all operations. If you modify values in these fields, the operational binding configuration with the **DEFAULT** setting will then contain the newly specified value.

  **Important:** When you change a field here at the service level, the change is propagated to the **DEFAULT** values in the operational bindings.

- **Format**
  
  You can select either the XML, JSON, or DYNAMIC format.

  This setting specifies the required format of the REST 2.0 requests and responses. See **Input argument encoding** and **Output argument encoding**, below.

- **HTTP Action**
  
  You can select the HTTP GET, POST, PUT, or DELETE action for submitting a request to an information service provider.

  This setting specifies the HTTP method that must be used when calling the service. See **HTTP Methods**, below.

- **Parameter Type**
  
  You can select BODY, URLEXTENDED, or URLQUERY.

  This setting specifies how service input argument values are encoded in the REST request. See **Input argument encoding** and **Output argument encoding**.

- **Security**
  
  You need to specify the security constraints for each service. The REST 2.0 binding uses these constraint definitions to properly define security constraints on the generated servlet and the corresponding service operation invocation URLs.

  If authentication is required, HTTP basic authentication is used. If confidentiality is required, HTTPS is used and must be configured in the application server.
Requires Authentication
Specifies if authentication is required. If you select this option, HTTP basic authentication is used for the generated servlet.

Requires Confidentiality
Specifies if data confidentiality is required for service invocations. If you select this option, HTTPS is required to invoke any of the service operation.

- REST 2.0 Settings in the pane for an operation.

In the REST 2.0 Settings area, you can use default values or specify another value for this particular operation.

Note: You set the default values at the Services level, in the Operation Bindings Default Settings area. When one of the default fields are changed at the service level, the value is automatically changed in the operational binding.

Format
You can select either DEFAULT, XML, JSON, or DYNAMIC format.

HTTP Action
You can select the DEFAULT, HTTP GET, POST, PUT, or DELETE action for submitting a request to an information service provider.

Parameter Type
You can select DEFAULT, BODY, URLEXTENDED, or URLQUERY.

- Operation Context
The default setting is <service-name>/<operation-name>. If you change the service name or operation name, the Operation Context value reflects the changes. This value will become a part of the URL that is used to invoke the service operation. The Operation Context value can be overwritten in the REST 2.0 Settings operation binding properties. See Calling the service by using REST, below.

For each service operation, the combination of its operation context (default or custom) and its HTTP method must be unique across all operations of all services contained in the application package (in the EAR). As a result, it is possible for two service operations to be invoked by using the same URL, provided that a different HTTP method is used for each of them. For example, The getCustomer operation is exposed by using the /marketing/customers/customer URL and a GET HTTP method and the deleteCustomer operation is exposed using the same /marketing/customers/customer URL and a DELETE HTTP method.

Restriction: There is a restriction if you use the same operation context (same URL) to invoke different operations. If a given operation context is used with an HTTP PUT or DELETE method, there can be only one other operation using the same operation context and it must use the HTTP GET method. Because HTTP PUT and DELETE methods can be both tunneled through an HTTP POST by simply using the X-HTTP-Method-Override as part of the HTTP request, it might become difficult to identify the correct method to call, and there might be security issues.

- Pass-through mechanisms
The REST 2.0 binding supports a pass-through mechanism both for input and output arguments. Using this mechanism, the service operation can receive the content of the HTTP request body “as is” and perform its own decoding logic and it can perform its own encoding logic on the result and pass it back as the content of the HTTP response body.

Input pass through
If you select this option, the service operation must have only one string input argument, the Parameter Type set to BODY and the associated HTTP method must be POST or PUT. In this case, the content of the HTTP request body is provided “as is” (no decoding or pass-through from the REST 2.0 binding perspective) to the service operation string input argument.

Output pass through
If you select this option, the service operation must return only one string value. The
associated HTTP method can be GET, POST, PUT or DELETE. In this case, the content of the HTTP response body is directly set by using the service operation returned string value (no encoding, pass-through from the REST 2.0 binding point of view). The Content-Type header of the HTTP response is set according to the Format configuration:

- Set to application/xml if the Format value is XML. The service operation is assumed to return an XML document.
- Set to application/json if the Format value is JSON. The service operation is assumed to return an JSON string.
- Set to application/xml or application/json if the Format value is DYNAMIC. The value of the Accept header is checked first. If application/xml and application/json are missing from the Accept header, then check the first character of the returned string: if it is a "less than" character, <, then application/xml is used, if it is an open curly brace, {, then application/json is used.

Input argument encoding

The encoding of the service operation input arguments is driven by two operation level REST 2.0 properties: Parameter Type and Format.

Parameter Type specifies which part of the HTTP request will hold the input arguments. Depending on the Parameter Type value, the Format value might then be used to further specify how the input arguments must be encoded. Valid values for Parameter Type are: URLQUERY, URLEXTENDED, BODY or MULTIPART.

URLQUERY

URLQUERY is supported only for GET and DELETE HTTP methods and only if all the service input arguments are scalar values (integer, float, boolean, string, etc). The service input arguments must be encoded in the URL query string using the standard URL parameter encoding mechanism:

http://servername:port/wisd-rest2/<application-name>/<OperationContext>?arg1=value1 &arg2=value2

The name of the URL parameters (arg1, arg2 in the example above) must match the name of the corresponding operation input argument.

URLEXTENDED

URLEXTENDED is supported only for GET and DELETE HTTP methods and only if all the service input arguments are scalar values (integer, float, boolean, string, etc). The service input arguments must be provided as part of the URL extended path:

http://servername:port/wisd-rest2/<application-name>/<OperationContext>/arg1Value/arg2Value

As the argument names are not specified, the arguments must be provided in the same order as defined in the operation signature.

BODY

BODY is supported only for POST and PUT HTTP methods and for any kind (scalar or complex) and number of service input arguments. The service input arguments must be provided in the HTTP request body and encoded in JSON or XML according to the Format value. See XML format and JSON format, below.

MULTIPART

MULTIPART is supported only for POST and PUT HTTP methods and for service operations with scalar only input arguments. The HTTP request must provide one part for each input argument with a name – name attribute of the Content-Disposition header of the part, for example, Content-Disposition: form-data; name="arg1" – matching the name of the corresponding operation input argument. Each part must contain the scalar value “as is” (no XML or JSON encoding).
Input argument format

The operation-level Format property that you configure in REST 2.0 Settings specifies the input argument encoding format for HTTP POST and PUT requests with a BODY parameter type as the arguments must be provided as part of the HTTP body. See Operation Bindings Default Settings, above.

XML  If XML is used, the input arguments must be XML encoded and supplied in the HTTP request body.

JSON  If JSON is used, the input arguments must be JSON encoded and supplied in the HTTP request body.

DYNAMIC  If Dynamic is used, the format can be different at each service invocation. The input arguments must be encoded in XML or JSON following the same rule as defined in the XML Format and JSON Format sections and the Content-Type header of the HTTP request must be set accordingly to application/xml or application/json. The REST 2.0 binding uses the Content-Type header value to determine the expected encoding format.

Output argument encoding

The encoding of the service operation output argument is driven by the operation level Format property. See Operation Bindings Default Settings, above. The output argument encoding is independent from the HTTP method that is used for the HTTP request (GET, POST, PUT or DELETE). The encoded output argument is supplied as part of the HTTP response body.

Output argument Format values

The operation-level Format property you configure in REST 2.0 Settings specifies the output arguments encoding format. Valid values for Format are: XML, JSON, and DYNAMIC.

XML  If XML is used, the output argument is XML-encoded and returned in the HTTP response body. The HTTP response Content-Type header is also set to application/xml.

JSON  If JSON is used, the output argument is JSON-encoded and returned in the HTTP response body. The HTTP response Content-Type header is also set to application/json.

DYNAMIC  If Dynamic is used, the client must include the application/xml or application/json content type in the Accept header of the HTTP request. The REST 2.0 binding uses the Accept header value to figure out how the HTTP response must be encoded and to properly set the Content-Type header. If the Accept header is missing or does not contain a valid content type (either application/xml or application/json), an HTTP Status 400 error is returned.

HTTP methods

Each service operation is associated with only one HTTP method. Clients must use the specified HTTP method in order to be able to call the corresponding service operation.

Due to the parameter encoding limitations of the GET and DELETE HTTP methods (parameters must be encoded as part of the URL), these methods can only be used for service operation without input arguments or with only scalar input arguments.

For service operations associated with PUT and DELETE HTTP methods, clients can use an HTTP POST to invoke these operations providing that they set the X-HTTP-Method-Override header of the HTTP request to the intended HTTP method (PUT or DELETE). This PUT/DELETE tunneling mechanism through POST is always enabled. For any HTTP POST request, if the X-HTTP-Method-Override header is set, then the HTTP method overwrite will take place. If the X-HTTP-Method-Override header is set to something other than PUT or DELETE, an HTTP 400 error (SC_BAD_REQUEST) is returned.
Calling the service by using REST

After you deploy your service, the service can be called by using the proper URL format and argument encoding.

URL Format

Each service operation is uniquely identified through the REST 2.0 binding by its URL and the HTTP method it is associated with. By default, the URL used to invoke each service operation is as follows:

http://servername:port/wisd-rest2/<application-name>/<OperationContext>

Note: OperationContext defaults to <service-name>/<operation-name> unless overwritten in the REST 2.0 Settings operation binding properties

XML format

Service requests encoded in XML are composed as follows:

- An XML root element enclosing all the input arguments (if any). The name of this root element can be anything. The REST 2.0 binding does not enforce any constraints on the name of the root element.
- Each input arguments (if any) are encoded using an XML subelement of the root element and named after the name of one of the input arguments of the operation.

Service responses encoded in XML are composed as follows:

- An XML root element called “results” directly containing the operation returned value.
- If the service operation does not return anything (void), an empty XML results element is used.
- The operation returned values is directly enclosed in the results root element. For example, if the operation returns a scalar value, an XML text node with that value is directly enclosed under the results root element.

Scalar data types

Scalar data types, or Java primitive types, are all encoded as XML text nodes (strings in the XML document). The value of the XML text node is the value returned by the Java toString() method for each of the primitive types. The only exception is for the java.util.Calendar type: the text node value is the date formatted in the ISO 8601 format: yyyy-MM-dd'T'HH:mm:ss.SSSZ.

Complex data types

For complex data types, a JavaBean instance is encoded into an XML chunk with each property (attribute) of the JavaBean encoded as an XML element in that chunk and named after the JavaBean property name.

Arrays

Java arrays are encoded as XML unbounded sequences. By default, there is enclosing XML element generated to hold all the array items. Array items are encoded like any other non array JavaBean attribute or service operation input argument, except that there is one XML element for each item of the array.

JSON format

Service requests encoded in JSON are composed of a single JSON object with one attribute for each service operation input argument. The name of these attributes must match the name of one of the input arguments of the operation. If the service operation does not have any input arguments, an empty JSON object must be provided: {}.

Service responses encoded in JSON are composed of a single JSON object with a single attribute called “results” holding the service operation returned value. If the service operation returns void, a null results JSON object is returned: {“results”:null}.
Scalar data types
The following table shows the list of supported scalar data types and the corresponding JSON format.

Table 6. Scalar data types and JSON format

<table>
<thead>
<tr>
<th>Scalar Data Type</th>
<th>JSON Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true or false</td>
</tr>
<tr>
<td>byte</td>
<td>Number</td>
</tr>
<tr>
<td>double</td>
<td>Number</td>
</tr>
<tr>
<td>float</td>
<td>Number</td>
</tr>
<tr>
<td>int</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.Boolean</td>
<td>true or false</td>
</tr>
<tr>
<td>java.lang.Byte</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.Double</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.Float</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.Integer</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.Long</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.Short</td>
<td>Number</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>String</td>
</tr>
<tr>
<td>java.math.BigDecimal</td>
<td>Number</td>
</tr>
<tr>
<td>java.math.BigInteger</td>
<td>Number</td>
</tr>
<tr>
<td>java.util.Calendar</td>
<td>Object representing a JavaScript Calendar:</td>
</tr>
<tr>
<td></td>
<td>{'date':26,'day':4,'hours':15,'minutes':38,'month':1,'seconds':6,'time':1235659086302,</td>
</tr>
<tr>
<td></td>
<td>'timezoneOffset':-60,'year':109}</td>
</tr>
<tr>
<td>long</td>
<td>Number</td>
</tr>
<tr>
<td>short</td>
<td>Number</td>
</tr>
</tbody>
</table>

Complex data type
A JavaBean instance is encoded into a JSON object with each property (attribute) of the JavaBean encoded as an attribute of the JSON object and named after the JavaBean property name.

Arrays
Java arrays are encoded as JSON arrays. Each element of the array is encoded as specified in Table 1, above. There is one exception to this rule for byte arrays (byte[]). In this case, the byte array is base64-encoded and is handled as a JSON String.

Error handling
By default, if any Java exception is thrown by the service operation being invoked, the REST 2.0 binding returns an HTTP 500 (Internal Server Error) response with the exception details encoded in the HTTP response body in XML or JSON format.

An example XML exception is:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<exception xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:java="http://java.sun.com">
<errorcode>[error code]</errorcode>
```
An example JSON exception return is:
{
    "results": {
        "errorcode": "[error code]",
        "stacktrace": "[The exception stack trace]",
        "classname": "[The exception class name]",
        "requestURI": "[The URI of the original request]"
    }
}

### Adding operations

Add operations to define the information tasks of a service. Each information service is composed of a set of operations that use artifacts such as database queries for its business logic. The implementation of operations, such as database queries and IBM InfoSphere DataStage jobs, is hidden from the service client. This loose coupling enables service clients to be developed independently of information services, and requires minimal coordination between the service developer and the service client developer.

### Before you begin

- "Creating a project" on page 30
- "Creating an application" on page 31
- "Creating a service" on page 33

### About this task

IBM InfoSphere Information Services Director supports the following operations providers:
- InfoSphere DataStage
- IBM InfoSphere QualityStage
- IBM DB2
- IBM InfoSphere Federation Server
- IBM InfoSphere Classic Federation Server for z/OS
- Oracle Database Server

### Procedure

1. In the Select a View pane, select the service that you want to add an operation to.
2. Select New > Operation.
3. In the Name field, enter the name of the operation.
4. Click Select in the Information Provider pane.
5. Select a type in the Select an Information Provider window. Refer to the Related Task that corresponds to your choice.
Related tasks:

“Creating a project” on page 30
A project is a collaborative environment that contains applications, services, and operations. You can define some access rights at the project level such as the users who can design or deploy applications.

“Creating an application” on page 31
An application is a container for a set of services. Because the application is the unit of deployment, all services within a single application must be deployed or undeployed together.

“Creating a service” on page 33
Create a service to group related operations. When you create a service, you attach bindings based on the method you use to invoke operations from your client. This task also contains a step to indicate your security requirements for the service.

“Chapter 2, “Supported information service providers,” on page 5
Information service providers are the sources of operations for your services. Using IBM InfoSphere Information Services Director, you can create services from the following information service providers: IBM InfoSphere DataStage; IBM InfoSphere QualityStage; IBM DB2 for Linux, UNIX, and Microsoft Windows; IBM InfoSphere Federation Server; IBM InfoSphere Classic Federation Server for z/OS; and Oracle Database Server.

Adding a DataStage and QualityStage type
Complete this task to expose an IBM InfoSphere DataStage job or an IBM InfoSphere QualityStage job as a service.

Before you begin

To use an InfoSphere DataStage job as a service, the property Enabled for Information Services must be selected in IBM InfoSphere DataStage and QualityStage Designer job properties. If you do not select this job property, the job will not be available in IBM InfoSphere Information Services Director.

Make sure the column names in your InfoSphere DataStage and IBM InfoSphere QualityStage jobs do not contain a numeric digit followed by lower case letter, such as 2b. This type of column name will cause your jobs to fail when you invoke them as IBM InfoSphere Information Services Director services.

Procedure

1. In the Select an Information Provider window, select DataStage and QualityStage type.
2. In the Select a DataStage Job pane, click the InfoSphere DataStage information provider that contains the job that you want.
3. Click the project to expand it.
4. Click a folder to expand it until you find the job that you want to use.
5. Select the job that you want to invoke.
6. Click OK.
7. View and edit inputs, outputs, and provider properties in the Information Provider pane.
Related concepts:

“Operation inputs, outputs, and provider properties” on page 62

After choosing the job which will define the operation for your service, you can edit the inputs, outputs, and provider properties in the Information Provider pane. (For Federated queries, this pane also includes an SQL Statement tab.)

Chapter 3, “Designing IBM InfoSphere DataStage and QualityStage jobs as services,” on page 7

Exposing IBM InfoSphere DataStage jobs and IBM InfoSphere QualityStage jobs as services implies a set of constraints and guidelines. The service-oriented architecture (SOA) platform supports three job topologies for different load and work style requirements: batch jobs, batch jobs with a service output stage, and jobs with service input and output stages.

Adding a DB2 or Federation Server type

Complete this task to expose a query or a stored procedure as a service from IBM DB2 or IBM InfoSphere Federation Server.

Before you begin

“Creating a project” on page 30

“Creating an application” on page 31

“Creating a service” on page 33

“Adding operations” on page 57

Procedure

1. Select DB2 or Federation Server type.
2. Choose a subtype from the menu.

<table>
<thead>
<tr>
<th>Subtype</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you choose SQL Statement</td>
<td>Choose Create SQL Statement to type a query or build one with the SQL Builder, or choose Browse Databases to select a nickname, table, or view.</td>
</tr>
<tr>
<td>If you choose Stored Procedure</td>
<td>1. Select a database. If you select an invalid data store, an invalid entry message displays. 2. Select a stored procedure to expose the details and arguments for that procedure. 3. Identify argument types that will define the operation inputs and outputs and their data types.</td>
</tr>
<tr>
<td>If you choose XML Source</td>
<td>XML Source may not be your best choice as an information provider subtype for users of DB2 9.1 or later. For handling TrueXML data in DB2, select the SQL Statement subtype and enter the required text to create a query. When using the TrueXML feature, XML-typed columns are treated in the same way as other database columns. However, the SQL Builder does not support the creation of XQuery or SQLXML queries.</td>
</tr>
</tbody>
</table>

3. Click OK.
4. View and edit inputs, outputs, the SQL statement, and provider properties in the Information Provider pane.
After choosing the job which will define the operation for your service, you can edit the inputs, outputs, and provider properties in the Information Provider pane. (For Federated queries, this pane also includes an SQL Statement tab.)

When you select a stored procedure as an operation in your service, you can view the metadata and default data types for the procedure.

**Adding an Oracle Database Server type**

Complete this task to expose a query or a stored procedure as a service from Oracle Database Server.

**Before you begin**

- "Creating a project" on page 30
- "Creating an application" on page 31
- "Creating a service" on page 33
- "Adding operations" on page 57

**Procedure**

1. Select Oracle Database Server type.
2. Choose a subtype.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you choose SQL Statement</td>
<td>Choose <strong>Create SQL Statement</strong> to type a query or build one with the SQL Builder, or choose <strong>Browse Databases</strong> to select a table or view.</td>
</tr>
</tbody>
</table>
| If you choose Stored Procedure | 1. Select a database.  
2. Select a stored procedure to expose the details and arguments for that procedure.  
3. Identify argument types that will define the operation inputs and outputs and their data types. |

3. Click **OK**.
4. View and edit inputs, outputs, the SQL statement, and provider properties in the Information Provider pane.

**Related concepts:**

- "Operation inputs, outputs, and provider properties" on page 62

Adding a Classic Federation Server type

Complete this task to expose a query or a stored procedure as a service from IBM InfoSphere Classic Federation Server for z/OS.
Before you begin

“Creating a project” on page 30
“Creating an application” on page 31
“Creating a service” on page 33
“Adding operations” on page 57

About this task

Refer to the SQL Reference for Classic Federation and Classic Data Event Publishing available from IBM support for details on using IBM InfoSphere Classic Federation Server for z/OS as an information provider. You can also refer to the IBM WebSphere Classic information center.

Procedure

1. Select Classic Federation Server type.
2. Choose a subtype.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you choose SQL Statement</td>
<td>Choose Create SQL Statement to type a query or build one with the SQL Builder, or choose Browse Databases to select a table or view.</td>
</tr>
</tbody>
</table>
| If you choose Stored Procedure | 1. Select a database.  
                              | 2. Select a stored procedure to expose the details and arguments for that procedure.  
                              | 3. Identify argument types that will define the operation inputs and outputs and their data types. |

3. Click OK.
4. View and edit inputs, outputs, the SQL statement, and provider properties in the Information Provider pane.

Related concepts:

“Operation inputs, outputs, and provider properties” on page 62

After choosing the job which will define the operation for your service, you can edit the inputs, outputs, and provider properties in the Information Provider pane. (For Federated queries, this pane also includes an SQL Statement tab.)

Stored procedures as operations

When you select a stored procedure as an operation in your service, you can view the metadata and default data types for the procedure.

Depending on whether you are using IBM DB2 stored procedures or federated stored procedures there may be some information required for defining stored procedures. A federated stored procedure is a federated database object that references a procedure on a remote data source that represents a repeatable and often used database task. When you create a federated procedure, the data types for the parameters for the data source procedure are mapped to the federated data types using the default forward data type mappings.

IBM InfoSphere Information Services Director allows you to browse through stored procedures that are already available in IBM DB2 or IBM InfoSphere Federation Server. InfoSphere Information Services Director will display metadata about these stored procedures:
• Shows how many result sets a stored procedure returns. InfoSphere Information Services Director does not support multiple result sets or stored procedures that require output parameters and a result set.
• No metadata about the result set itself.

If a stored procedure returns a result set, remember to check Return Array, found in the Information Provider tab under the Output tab.

Related tasks:
"Adding a DB2 or Federation Server type" on page 59
Complete this task to expose a query or a stored procedure as a service from IBM DB2 or IBM InfoSphere Federation Server.

---

**Operation inputs, outputs, and provider properties**

After choosing the job which will define the operation for your service, you can edit the inputs, outputs, and provider properties in the Information Provider pane. (For Federated queries, this pane also includes an SQL Statement tab.)

You can browse through the Inputs, Outputs and Provider Properties tabs to review input and output parameters for the service.

In the Inputs tab, you can see the names of the input arguments and their type. Typically you will change the argument names from the defaults generated by IBM InfoSphere Information Services Director to something more meaningful to a developer. For example, you might change arg1 to myjob. If you do not want to change each of the argument names, you can select Set Constant and assign a value or name to the set of arguments and their types. If you select Set Constant and do not enter a value, the operation fails.

Looking at the Outputs tab, you see the outputs returned by the operation. You can also choose whether the result of the operation is a single row or an array. Also, fields can be concatenated into a single structure, which requires you to specify the data type and variable name for that structure.

Provider properties enable you to set runtime parameters such as the user credentials used to run the operation. You can change these settings during runtime. Refer to the runtime settings topic for more details. You can choose design time default values for the following fields:
• Active Job Instances or JDBC Connections
• Idle Time
• Activation Threshold
• Maximum Run Time
• Stuck Timeout
• Request Link
• Credentials
Related concepts:
“Runtime settings” on page 66
During the design of the service operation, default settings for runtime execution are created.

Related tasks:
“Adding a DataStage and QualityStage type” on page 58
Complete this task to expose an IBM InfoSphere DataStage job or an IBM InfoSphere QualityStage job as a service.

“Adding a DB2 or Federation Server type” on page 59
Complete this task to expose a query or a stored procedure as a service from IBM DB2 or IBM InfoSphere Federation Server.

“Adding a Classic Federation Server type” on page 60
Complete this task to expose a query or a stored procedure as a service from IBM InfoSphere Classic Federation Server for z/OS.

“Adding an Oracle Database Server type” on page 60
Complete this task to expose a query or a stored procedure as a service from Oracle Database Server.
Chapter 7. Deploying IBM InfoSphere Information Services Director applications

After the information service has been designed and saved, it must be deployed.

Procedure
1. To publish the application you have just designed, select an application in the Select Information Services Application to Work With object list.
2. Click Deploy from the Tasks pane.
3. Optional: In the application pane that opens, you can exclude one or more services, bindings, or operations from the deployment, change runtime properties such as minimum number of job instances, or, for IBM InfoSphere DataStage jobs, set constant values for job parameters.
4. Click Deploy at the bottom right of the application pane. When starting the deployment process, a status bar will move back and forth at the bottom of the pane. IBM InfoSphere Information Services Director deploys the enterprise archive (EAR) file on the application server.
5. Optional: Click Details to expand the Activity Status pane and watch the status of the deployment. The status changes to Completed after the deployment has finished.
6. Click Close.

Viewing deployed services in the Information Services Catalog tab of the IBM InfoSphere Information Server Web console

You can access information about deployed information services in the Information Services Catalog tab of the IBM InfoSphere Information Server Web console. For example, you can search for deployed services by custom attributes and design-time metadata. For a service with the SOAP over HTTP binding, you can verify deployment by opening the Web Service Definition Language (WSDL) document, which contains all the necessary descriptions (metadata) that a client application needs to invoke the service.

About this task

Restrictions
• To create categories and custom attributes for deployed services, you must have the role of Information Services Catalog Manager.

Procedure
1. From the IBM InfoSphere Information Server console, select Operate > Deployed Information Service Applications.
2. Expand a deployed application to view the services within it.
3. Select a service to view the description, bindings, and contact information.
4. Click View Service in Catalog.
5. From the Select a View pane, select Bindings.
6. If the SOAP Over HTTP binding is attached to your service, click Open WSDL Document to view the WSDL document.
7. If the EJB binding is attached to your service, click Download Client JAR File to download the client JAR file.
8. Click the Information Services Catalog link in the upper-right corner to view the full contents of the Information Service Catalog tab and perform other tasks.
About WSDL documents

The different sections within a WSDL document describe all necessary information about the service and all of its operations. The document contains information about the structure of input and output messages, the operation names, and the address at which the service is located, among other things.

IBM InfoSphere Information Services Director generates the WSDL document dynamically. The pattern to generate the document is as follows: http://[IS_HostName]:[IS_WAS_Port]/[WISD_Application_Name]/[WISD_ServiceName]/wsdl/[WISDService].wsdl

The WSDL file is automatically generated and displayed in a new browser window. To test the service, identify the location of the WSDL describing the deployed service. By copying the WSDL URL, you can use many different testing tools to invoke your Web service. Most tools prompt you for the URL of the WSDL document.

Runtime settings

During the design of the service operation, default settings for runtime execution are created.

These defaults can be set at runtime to adjust these values for deploying specific services. To meet ad hoc demands, adjust the runtime behavior of service operations and information providers.

The flow of requests involves interactions among the load balancer, operations queues, and pipelines. Each plays a key role:

- **Load balancer**: When there are multiple ASB Agents, the load balancer is used to determine to which ASB Agent a request will be routed.
- **Pipeline**: Some providers (IBM InfoSphere DataStage, for example) can process multiple requests at the same time. This is called pipelining. IBM InfoSphere Information Services Director takes advantage of that ability by defining the size of the pipeline using the provider's runtime parameter. When multiple pipelines exist on the same ASB Agent for an operation, InfoSphere Information Services Director will attempt to keep the same number of requests in each pipeline.
- **Operation queues**: When all pipelines are filled, the requests wait in what is called the operation queue. When defining an operation in the IBM InfoSphere Information Server console, you can specify the size of this queue and the maximum time a request can wait on the queue.

Active job instances or JDBC connections

Active job instances or connectors are the minimum and maximum number of jobs or connections that will be active at a time. The ASB Agent will attempt to always have at least the minimum active instances available. Also, the ASB Agent will attempt to reduce the number of active jobs or connections to that minimum amount. The ASB Agent will not start new instances after the maximum active limit has been reached. The default minimum is 0. The default maximum is 1.

Idle time

Idle time parameters determine the amount of time jobs or connections can be idle.

**Min Idle Time**
- After a job or connection has been idle for this amount of time, it is eligible to be stopped in order to reduce the total number of active jobs or connections to the Min Active amount. Therefore this value provides a way to keep the ASB Agent from starting and stopping jobs or connections too often. The default minimum is 1 second.

**Max Idle Time**
- This is the maximum time that a job or connection can be idle before it must be stopped. This value is mostly used when the underlying job or connection has known limits to how long it can be left running without activity. For example, if a database will automatically close a connection
that is idle for more than 5 minutes, it might be a good idea to set the max idle value to 240 seconds (4 minutes). The default maximum is 0 seconds.

**Operation queues parameters**

**Activation threshold**

Activation threshold sets the number of service requests in the operation queue. This number should be less than the maximum pipeline size. Activation Threshold has two settings.

**Maximum Service Requests**

Represents the number of requests that must be on the Operation Queue before a timer is started that will be used to decide if a new job or connection needs to be activated. It is important to understand that this value represents requests on the Operation Queue and does not include requests already in the pipeline. The default is 0.

**Maximum Delay**

The length of that time before a timer is started to decide if a new job or connection needs to be activated. When the timer expires, a new job or connection will be activated. If the number of requests in the operation queue goes below 3, the timer will be stopped and no new job or connection will be activated. The default is 0 milliseconds.

Consider the following two examples.

- An operation is attached to a DataStage job that should take about 0.5 seconds to process each request. The operation queue size is set to 50 and the wait time is set to 3000 (3 seconds). In this case, only the first six requests on the queue can make it to the pipeline; the seventh request will probably time out after the 3-second wait time. In this case the wait time can be increased or the queue size decreased, or both.

- An operation is attached to a DataStage job that should take about 20 milliseconds to process each request. The operation queue size is set to 10 and the wait time is set to 1000 (1 second). This is basically the opposite of the first example. In this case, nothing should ever time out because it will only take 200 milliseconds to clear a full queue. Increasing the max queue size probably makes sense in this case.

**Pipeline parameters**

**Maximum runtime**

The Maximum Run Time property sets the maximum lifetime of a job instance. If the limit is exceeded, the job instance continues to process the service requests that are present in the pipeline buffer. However, the job instance becomes unavailable to new service requests. If the job instance becomes idle because the pipeline is empty, the job instance is stopped. The default is 0 seconds.

**Request limit**

Within IBM InfoSphere Information Services Director you can take advantage of that by defining the size of the pipeline using the Request Limit property of a provider. When multiple pipelines exist on the same ASB Agent for an operation, InfoSphere Information Services Director will attempt to keep the same number of requests in each pipeline. The default is 0.

**Stuck timeout**

Stuck timeout provides a maximum for which the job should be terminated because it indicates a probable problem with the service. After this time is reached, the InfoSphere Information Services Director server terminates the service and sends an error code to client. The default is 0 seconds.

**Load balancing parameters**

The load balancing function determines which ASB Agent a request will be routed to. In InfoSphere Information Services Director you can choose between two load balancers: Round Robin and Average Response Time. Load balancing only attempts to achieve a balance between different ASB Agents.
Round Robin
This is the default and should probably always be used in a homogeneous environment. Each request is simply sent to the ASB Agents in order (for example, Agent-1, Agent-2, Agent-3, repeat). Even if an operation has four jobs on Agent-1 and one on Agent-2, the requests will go to Agent-1, then Agent-2, then Agent-1, and so on. The intent is to balance the work among the agents, not among the jobs. If four requests are sent to Agent-1 and only one to Agent-2, then the load on Agent-1 would be heavier than the load on Agent-2.

Average Response Time
In the average response time allocation, the average time among information providers in processing a number of service requests is calculated. The default number of requests is 100. A percentage is applied to each provider (job or federated object) that is eligible to implement the service operation. The percentage is based on the average response time from the provider in processing service requests. This function is likely more useful in a heterogeneous environment. It uses the response time of each request to help determine where requests should be routed. The system with the shorter response time will receive a higher percentage of the overall number of requests.

Related concepts:
- "Always-running instances" on page 14
- "Operation inputs, outputs, and provider properties" on page 62

Follow these guidelines for jobs that are always-running instances.

After choosing the job which will define the operation for your service, you can edit the inputs, outputs, and provider properties in the Information Provider pane. (For Federated queries, this pane also includes an SQL Statement tab.)

IBM InfoSphere Information Services Director service requests take a series of steps to be processed.

---

Publishing services to IBM WebSphere Service Registry and Repository (WSRR)

IBM WebSphere Service Registry and Repository (WSRR) is an application that supports annotation of services with information that is used to select, invoke, govern, and reuse services.

Before you begin

Make sure that you have completed the steps to integrate IBM WebSphere Service Registry and Repository (WSRR) with IBM InfoSphere Information Server. See Configuring IBM InfoSphere Information Server to integrate with WebSphere Service Registry.

About this task
- You can publish deployed IBM InfoSphere Information Services Director services to the WebSphere Service Registry and Repository application by using the Information Services Catalog in the IBM InfoSphere Information Server Web console to view or update configuration information.
- The following instructions assume that InfoSphere Information Server is installed in the C:\IBM\InformationServer directory.

The following procedure describes how to configure the Information Services Catalog to integrate with WebSphere Service Registry and Repository.

Procedure
1. Log in to the IBM InfoSphere Information Server Web console.
2. Click the Information Services Catalog tab.
3. In the Navigation pane, select Configure Service Registry.
4. In the **Configure Service Registry** tab, specify the service registry connection, undeployed services settings and IBM InfoSphere Information Server configuration information.

5. Configure the connection to WebSphere Service Registry and Repository by supplying the following service registry connection information:

   a. Specify the URL of the WSRRCoreSDOService that is the deployed WebSphere Service Registry Repository Web service. The value of the URL is specified in the WebSphere Service Registry Repository WSDL file. The default value of the URL is: 'http://registryHostName:Port/WSRRCoreSDO/services/WSRRCoreSDOPort'.

   b. Specify the user name. This field is required only if the security of the IBM WebSphere Application Server that hosts WebSphere Service Registry Repository is enabled.

   c. Specify the password. This field is required only if the security of the WebSphere Application Server hosts WebSphere Service Registry Repository is enabled.

   d. Optional: You can manage the deletion of undeployed services by specifying the expiration time. If you select **No Expiration**, undeployed services do not expire and remain in the list of undeployed services until deleted or re-deployed. If you want the undeployed service to expire, select **Expire after 0 Days** and enter the number of days after which you want the service to expire. An undeployed service that expired is automatically deleted when you click **Delete All Expired** in the **Manage Undeployed Services** tab.

6. Verify the connection with the WebSphere Registry Repository server.

7. To publish a service to WebSphere Service Registry Repository, click **Manage Deployed Service**.

8. In the **Manage Deployed Services** tab, select the service that you want to publish and click **Publish to Service Registry**.

---

## Configuring InfoSphere Information Server to integrate with WSRR

Before you can publish IBM InfoSphere Information Services Director services to the WebSphere Service Registry and Repository application, you need to configure IBM InfoSphere Information Server to integrate with WebSphere Service Registry and Repository (WSRR). This task describes how to configure the Information Services Catalog to integrate with WSRR.

### About this task

- You must do this task before you can publish deployed IBM InfoSphere Information Services Director services to the WebSphere Service Registry and Repository application.

- After you integrate WSRR with IBM InfoSphere Information Server, you can enable InfoSphere Information Server and WSRR to communicate through a security channel, by using Secure Sockets Layer (SSL).

- The following instructions assume that InfoSphere Information Server is installed in the `C:\IBM\InformationServer` directory.

### Procedure


2. Create the directory: `\IBM\InformationServer\ASBServer\profile\informationServer\wsrr`

3. Unzip the WSRR client file to: `C:\IBM\InformationServer\ASBServer\profile\informationServer\wsrr`

4. Run following command in `C:\IBM\InformationServer\ASBServer\bin` directory to propagate the `wsrr` client files:

   - For a cluster environment:
     ```
     FilePropagator.bat -user wasadmin -password wasadmin_password
     -srcDir C:\IBM/InformationServer/ASBServer/profile/informationServer/wsrr
     -destDir informationServer/wsrr
     ```
Note: The wasadmin user ID is the WebSphere Application Server administrator user ID and wasadmin_password is the password of wasadmin.

- For a stand-alone environment:
  FilePropagator.bat
  -srcDir C:/IBM/InformationServer/ASBServer/profile/informationServer/wsrr
  -destDir informationServer/wsrr

5. Configure the following settings in the WebSphere Application Server administrative console, for either a cluster or stand-alone environment:
   - For a cluster environment, use these steps:
     a. Click Servers > Clusters > Cluster01 > Cluster members
     b. For each server listed, do the following steps:
        1) Click the server. For example, server1
        2) Click Java and Process Management > Process Definition > Java Virtual Machine
        3) Add following to the Classpath field:
           ${USER_INSTALL_ROOT}/informationServer/wsrr/sdo-int.jar
        4) Click OK.

        Note: Repeat step b until you have made the change to all servers listed in Cluster members.
   - For a stand-alone environment, use these steps:
     a. Click Servers > Application servers > server1
     b. Click Java and Process Management > Process Definition > Java Virtual Machine
     c. Add following to the Classpath field:
        ${USER_INSTALL_ROOT}/informationServer/wsrr/sdo-int.jar
     d. Click OK.

6. Continue configuring settings in the WebSphere Application Server administrative console:
   a. Click Environment > Shared Libraries
   b. Select the scope to be Cell level
   c. Click New, then fill the form as follows:
      Name: wsrr_lib
      Classpath: informationServer/wsrr/ServiceRegistryClient.jar
   d. Click OK.
   e. Click Applications > Enterprise Applications > RTICatalog_svc.ear > Shared library references
   f. Select RTICatalog_svc.ear, then click Reference shared libraries
   g. Add wsrr_lib to Selected list.
   h. Click OK.

7. Restart the cluster (or the server for a stand-alone application server) to make changes effective.

What to do next

If you want InfoSphere Information Server and WSRR to communicate through a security channel, you can configure InfoSphere Information Server to communicate with WSRR through SSL.

Configuring InfoSphere Information Server to communicate with WSRR through SSL

You can enable InfoSphere Information Server and WSRR to communicate through a security channel by using Secure Sockets Layer (SSL).
Before you begin

Make sure that you have completed the steps to integrate IBM WebSphere Service Registry and Repository (WSRR) with IBM InfoSphere Information Server. See Configuring IBM InfoSphere Information Server to integrate with WebSphere Service Registry.

About this task

- You can do these steps if you want InfoSphere Information Server and WSRR to communicate through a secure channel. This is not a requirement for publishing.
- The following instructions assume that InfoSphere Information Server is installed in the C:\IBM\InformationServer directory.

Procedure

1. Copy the signer certificate of the WSRR system to a temporary directory, for example, C:\temp\wsrrcert.arm
2. Log in to IBM WebSphere Application Server administrative console and configure the settings that are specific to your environment (either cluster or stand-alone):
   - For a cluster application server configuration:
     a. Click Security > SSL certificate and key management > Key stores and certificates > CellDefaultTrustStore > Signer certificates
     b. Click Add, then fill the form as:
        - Alias: wsrr
        - File name: C:\temp\wsrrcert.arm
        - Data type: Base64-encoded ASCII data
     c. Click OK.
     d. Click Security > SSL certificate and key management > SSL configurations > CellDefaultSSLSettings
     e. Verify that the trust store name is: CellDefaultTrustStore.
     f. Click Servers > Clusters > Cluster01 > Cluster members
     g. For each server that is listed, do the following:
        - Click the server. For example, server1.
        - Click Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties
        - Click New, then complete the form as:
          - Name: ssl.configName
        - Value: CellDefaultSSLSettings
   - For a stand-alone application server:
     a. Click Security > SSL certificate and key management > Key stores and certificates > NodeDefaultSSLSettings > Signer certificates.
     b. Click Add, then fill the form as:
        - Alias: wsrr
        - File name: C:\temp\wsrrcert.arm
        - Data type: Base64-encoded ASCII data
     c. Click OK.
     d. Click Security > SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings
     e. Verify that the trust store name is: NodeDefaultSSLSettings.
     f. Click Servers > Application servers > server1.
g. Click **Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**

h. Click **New**, then fill the form as:
   - Name: ssl.configName
   - Value: NodeDefaultSSLSettings

i. Click **OK**.

3. Now restart the cluster (or the server for standalone application server) to make the change effective.

**Related information:**
- [Restarting application server processes in a cluster](#)
- [Administering IBM InfoSphere Information Server and IBM WebSphere Application Server services](#)
- [Enablement of Secure Sockets Layer (SSL)](#)
Chapter 8. Service consumers

Service consumers are able to access information services using multiple technologies for program interoperability.

Related concepts:

You can more easily create business processes with information from a complex, heterogeneous environment if that information is published as consistent and reusable services.

IBM WebSphere Process Server as a service consumer

After you deploy an information service, it can be invoked by IBM WebSphere Process Server through any of the supported bindings such as SOAP over HTTP or EJB.

For WebSphere Process Server applications, an interface has been developed to allow users to employ information services as part of orchestrations defined in IBM WebSphere Integration Developer. By invoking an information service, you can use the information in a business process in various ways. You can, for example:

- Guide the control flow. For example, you might want to use inventory information to decide if a supplier service needs to be called for items that will shortly be out of stock.
- Retrieve business data and send it, for example, to partners, clients, or other services.
- Update business data. For example, the delivery time of an item can be changed based on the information received from the supplier of this item.

In the following example, a WebSphere Integration Developer user can define a business integration module in which the information service delivers data to the process. The first step is to insert a new information service activity into the business process. The user then discovers an appropriate information service to bind to the new activity. In the Information Service Operation window, the user identifies the IBM InfoSphere Information Services Director server in which the services are cataloged and deployed.

You can browse the contents of the metadata server in IBM InfoSphere Information Server to obtain details about available services. The browser will enable the user to discover deployed services and operations. After you have selected the service, you can view detailed additional information about the selected operation, including implementation details. The input and output parameters for the selected information service are then mapped to business process variables. Complex data types that are imported through the operation selection are automatically added to the list of selectable data types.

After the business module containing the information service has been added to the WebSphere Process Server configuration, information service runtime support within WebSphere Process Server enables it to invoke the new process.

SCA support in IBM InfoSphere Information Server

Service Component Architecture (SCA) is a set of specifications that describes a model for building applications and systems using a service-oriented architecture (SOA).

SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services. SCA encourages an SOA organization of business application code based on components that implement business logic, which offer their capabilities through service-oriented interfaces and which consume functions offered by other components through service-oriented interfaces, called service references.
The IBM WebSphere Enterprise Service Bus (ESB) uses SCA to define its interfaces to its mediation flows. The Enterprise Service Discovery wizard is a tool that guides the ESB developer through the discovery of existing services. IBM InfoSphere Information Services Director provides an extension to the Enterprise Service Discovery wizard that allows a user to browse a InfoSphere Information Services Director service repository. This extension shows some of the specialized metadata of the information service, including operational metadata. Based on this metadata, you can then decide which service fits best. For each operation to be imported, a Web service import binding will be created in the assembly editor.

The InfoSphere Information Services Director Enterprise Service Discovery extensions have the same look and feel as other service providers.

Selecting a service comprises the following steps (each step is represented by a separate page in the Enterprise Service Discovery wizard):

1. Open the wizard and select the InfoSphere Information Services Director service provider.
2. Connect to a InfoSphere Information Services Director instance by providing server name, ports, user ID and password.
3. Browse a tree of InfoSphere Information Services Director applications, projects, services, operations, and operation parameters. For operations, click the Details button in the wizard to view metadata that contain, for example: service description, date created, IBM IBM InfoSphere DataStage job reports (if applicable), and federated query of an II service (if applicable). It is also possible to filter according to operation types (InfoSphere DataStage and WII, for example).
4. Select operations to import, optionally selecting the subset of bindings for which the import should be done.
5. Select the SCA module where the service should be imported.

After all artifacts have been created, the assembly editor can be used in the usual way to integrate the new Web service import binding in the SCA module.
Appendix A. Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible. The installation program installs the following product modules and components:

- IBM InfoSphere Business Glossary
- IBM InfoSphere Business Glossary Anywhere
- IBM InfoSphere DataStage
- IBM InfoSphere FastTrack
- IBM InfoSphere Information Analyzer
- IBM InfoSphere Information Services Director
- IBM InfoSphere Metadata Workbench
- IBM InfoSphere QualityStage

For information about the accessibility status of IBM products, see the IBM product accessibility information at http://www.ibm.com/able/product_accessibility/index.html.

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to set display preferences in your browser. It also allows you to use screen readers and other assistive technologies to access the documentation.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.
Appendix B. Reading command-line syntax

This documentation uses special characters to define the command-line syntax.

The following special characters define the command-line syntax:

[ ] Identifies an optional argument. Arguments that are not enclosed in brackets are required.

... Indicates that you can specify multiple values for the previous argument.

| Indicates mutually exclusive information. You can use the argument to the left of the separator or
the argument to the right of the separator. You cannot use both arguments in a single use of the
command.

{} Delimits a set of mutually exclusive arguments when one of the arguments is required. If the
arguments are optional, they are enclosed in brackets ([ ]).

Note:

- The maximum number of characters in an argument is 256.
- Enclose argument values that have embedded spaces with either single or double quotation marks.

For example:

**wsetsrc** [-S server] [-l label] [-n name] source

The source argument is the only required argument for the wsetsrc command. The brackets around the
other arguments indicate that these arguments are optional.

**wlsac** [-l | -f format] [key... ] profile

In this example, the -l and -f format arguments are mutually exclusive and optional. The profile argument
is required. The key argument is optional. The ellipsis (...) that follows the key argument indicates that you
can specify multiple key names.

**wrb -import** {rule_pack | rule_set}...

In this example, the rule_pack and rule_set arguments are mutually exclusive, but one of the arguments
must be specified. Also, the ellipsis marks (...) indicate that you can specify multiple rule packs or rule
sets.
Appendix C. Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

*Table 7. IBM resources*

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description and location</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Support Portal</td>
<td>You can customize support information by choosing the products and the topics that interest you at <a href="http://www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server">www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server</a></td>
</tr>
<tr>
<td>Software services</td>
<td>You can find information about software, IT, and business consulting services, on the solutions site at <a href="http://www.ibm.com/businesssolutions/">www.ibm.com/businesssolutions/</a></td>
</tr>
<tr>
<td>My IBM</td>
<td>You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at <a href="http://www.ibm.com/account/">www.ibm.com/account/</a></td>
</tr>
<tr>
<td>Training and certification</td>
<td>You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at <a href="http://www.ibm.com/software/sw-training/">http://www.ibm.com/software/sw-training/</a></td>
</tr>
<tr>
<td>IBM representatives</td>
<td>You can contact an IBM representative to learn about solutions at <a href="http://www.ibm.com/connect/ibm/us/en/">www.ibm.com/connect/ibm/us/en/</a></td>
</tr>
</tbody>
</table>
Appendix D. Accessing and providing feedback on the product documentation

Documentation is provided in a variety of locations and formats, including in help that is opened directly from the product client interfaces, in a suite-wide information center, and in PDF file books.

The information center is installed as a common service with IBM InfoSphere Information Server. The information center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open the information center from the installed product or from a Web browser.

Accessing the information center

You can use the following methods to open the installed information center.

- Click the Help link in the upper right of the client interface.

  Note: From IBM InfoSphere FastTrack and IBM InfoSphere Information Server Manager, the main Help item opens a local help system. Choose Help > Open Info Center to open the full suite information center.

- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in Web clients.

- Use a Web browser to access the installed information center even when you are not logged in to the product. Enter the following address in a Web browser: http://host_name:port_number/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.home.doc/ic-homepage.html. The host_name is the name of the services tier computer where the information center is installed, and port_number is the port number for InfoSphere Information Server. The default port number is 9080. For example, on a Microsoft® Windows® Server computer named iisdocs2, the Web address is in the following format: http://iisdocs2:9080/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.nav.doc/dochome/iisinfsrv_home.html.

A subset of the information center is also available on the IBM Web site and periodically refreshed at http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r7/index.jsp

Obtaining PDF and hardcopy documentation

- A subset of the PDF file books are available through the InfoSphere Information Server software installer and the distribution media. The other PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&cwv=1

- You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss

Providing comments on the documentation

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- To comment on the information center, click the Feedback link on the top right side of any topic in the information center.

- Send your comments by using the online readers’ comment form at www.ibm.com/software/awdtools/rcf/
• Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, include the location of the text (for example, a title, a table number, or a page number).

• You can provide general product feedback through the Consumability Survey at [www.ibm.com/software/data/info/consumability-survey](http://www.ibm.com/software/data/info/consumability-survey)
Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 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 is for planning purposes only. The information herein is subject to change before the products described become available.

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. The sample programs are provided “AS IS”, without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.
If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACS\textsuperscript{Link}, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS\textsuperscript{Link} licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.
InfoSphere Classic Federation Server
operations 61
adding 61
InfoSphere DataStage 5
connecting to 25
job topologies 7
InfoSphere DataStage job operations 58
adding 58
InfoSphere DataStage jobs 11
always running instances 14
design features 12
designing services 7
global considerations 11, 12
multiple job instances 13
runtime constraints 12
InfoSphere Federation Server 5
connecting to 25, 26
InfoSphere Information Services Director
applications 65
architecture 2
configuration 2
deploying 65
infrastructure services 1
introduction 1
message flow 3
Two-source Match stage 16, 20, 21
InfoSphere Master Data Management Server 27
registering services 27
InfoSphere MDM Server 27
registering services 27
InfoSphere QualityStage 5
connecting to 25
job topologies 7
InfoSphere QualityStage job operations 58
adding 58
InfoSphere QualityStage jobs 7
always running instances 14
design features 12
global considerations 11, 12
multiple job instances 13
runtime constraints 12
infrastructure services
description 1
load balancing and availability services 1
Logging services 1
Security services 1
Service catalog 1
input and output links
Data Source stage 13
inputs
operations 62
introduction
InfoSphere Information Services Director 1
ISD Input stage
adding 16
configuring 16
timeout value 23
ISD Output stage
adding 15
configuring 15
timeout value 23
J
Java Message Service 35, 36, 37, 38, 39, 40, 41, 42
bindings 35, 36, 37, 38, 39, 40
JMS 35, 36, 37, 38, 39, 40
bindings 35, 36, 37, 38, 39, 40
JMS bindings
adding IS to a service integration bus 36
associating queues in a service integration bus 38
associating topics in a service integration bus 38
configuring 35
creating a connection factory 38
creating a service integration bus 36
creating an activation specification 39
defining queues in a service integration bus 37
setting up 40
setting up in WAS 36
job topologies
InfoSphere DataStage 7
InfoSphere QualityStage 7
L
legal notices 83
load balancer 3
load balancing and availability services 1
load balancing parameters
runtime settings 66
Logging services 1
M
macros and basic functions
global considerations 11
message flow 3
multiple job instances
Database stages 14
design considerations 13
FTP stage 14
InfoSphere DataStage jobs 13
InfoSphere QualityStage jobs 13
Sequential File stage 14
O
operations 29, 57, 61
adding 57
define service tasks 57
developing 29
editing 62
inputs 62
outputs 62
provider properties 62
stored procedures 61
operations queues parameters
runtime settings 66
Oracle Database Server 5
connecting to 25, 26
Oracle Database Server operations (continued)
adding 60
outputs
operations 62
P
product accessibility
accessibility 75
product documentation
accessing 81
projects 29, 30
about 30
creating 30
description 30
new 30
prerequisites 30
properties
SOAP Over JMS bindings 41
provider properties
operations 62
published services 68, 69, 71
publishing
services 68, 69, 71
R
real-time 20, 21
real-time jobs
Two-source Match stage 16, 18
registering services
InfoSphere Master Data Management Server 27
InfoSphere MDM Server 27
Reprensentational State Transfer bindings 50, 51
request and response formats
RSS bindings 50
REST 50
bindings 50
REST 2.0 51
bindings 51
REST 2.0 bindings 51
description 51
REST bindings 50
description 50
RSS 48, 49, 50
bindings 48, 49, 50
RSS bindings 49, 50
attaching 49
configuring 49
request and response formats 50
setting up 48
RTI export format files 32
RTI files
importing 32
runtime constraints
always running instances 14
InfoSphere DataStage jobs 12
InfoSphere QualityStage jobs 12
runtime settings
active job instances 66
description 66
load balancing parameters 66
operation queues parameters 66
Index 89

S
SCA support 73
service consumers 73
Security services 1
Sequential File stage
  multiple job instances 14
Service catalog 1
Service Component Architecture 73
  support 73
  service consumers
    about 73
    description 73
    SCA support 73
    WebSphere Process Server 73
service integration bus 36
  adding IS 36
  associating queues 38
  associating topics 38
  defining queues 37
service requests 3
processing 3
services 29, 33, 34, 35
  about 34
  accessing 35
  attaching a binding 35
  creating 33
  description 33, 34
  developing 29
  new 33
  prerequisites 33
  publishing 68, 69, 71
Services Catalog 65
services tier 2
services, publishing as
  Lookup stage 20
Two-source Match stage 16, 20, 21
setting up
  JMS bindings 40
  RSS bindings 48
SOAP Over HTTP 42
  bindings 42
SOAP Over HTTP bindings
description 42
SOAP Over JMS 40, 41
  bindings 40, 41
SOAP Over JMS bindings
  attaching 40
  properties 41
SOAP Over JMS bindings, attaching 40
software services
  contacting 79
special characters
  in command-line syntax 77
stored procedures
  operations 61
stored procedures as operations 61
  using 61
stream output link
  Data Source stage 13
  support
  customer 79
syntax
  command-line 77

T
Text over HTTP
  service binding 43
Text over HTTP bindings
description 43
Text Over JMS 42
  bindings 42
Text Over JMS bindings
  attaching 42
timeout value
  ISD Input stage 23
  ISD Output stage 23
trademarks
  list of 83
Two-source Match stage
  InfoSphere Information Services
    Director 16, 20, 21
    job design 17
    real-time 20, 21
    real-time jobs 16, 18
    services, publishing as 16, 20, 21
    Web services 18

W
WAS
  JMS bindings 36
Web services
  Two-source Match stage 18
WebSphere Application Server
  JMS bindings 36
WebSphere Process Server 73
  service consumers 73
WebSphere Service Registry and Repository 68, 69, 71
WSDL document
  using existing 43
  using pre-defined 43
WSDL documents
  about 66
  contents 66
  description 66

X
XSLT
  service bindings 43