Designing Integration with Adapter Development Kit

You can use the Service Link Adapter Development Kit (ADK) to develop Service Link adapters. The ADK is the set of components that allow the production of adapters for the Service Link subsystem of Service Catalog. Service Link provides external communications for Service Catalog and provides for coordinated externalization of workflow tasks with other systems.

To achieve this communication, Service Link supports installable adapters. Service Link ships with standard adapters, but developers can create other adapters. This chapter describes the process of writing adapters.

This chapter is intended for:

- **Administrator**. The administrator has access to the product packages and can install Service Catalog products in a customer system.
- **Adapter Developer**. The adapter developer is a person that is well versed in Java technologies, including ANT, and it is a subject matter expert of the integration required.

For more support on building custom adapter, contact *Cisco TAC*.

Getting Started

This section describes the installation of the ADK, its structure, compiling adapters, and adapter deployment.

Installing the JDK

Follow the instructions from Sun or IBM to install the Java Development Kit. Service Catalog is certified with Sun JDK 6 for installation on WebLogic 12.2.1.2 or JBoss 7.1.1, and with IBM Java 1.6 for installation on WebSphere 7.0.0.17.

Installing the ADK

To install the ADK:

1. **Administrator**: Expand the context of the product packages, locate the adk.zip under the image/isee/dist folder, and inform the adapter developer of the location.

2. **Adapter developer**: Obtain the file adk.zip (or adk.tar.gz) from the administrator.
3. **Adapter developer**: Expand the ADK package in a local machine, in C:\ADK. It does not have to be the C drive, nor the ADK directory. However, the examples in this chapter use C:\ADK.

ADK Structure

After installing the ADK, the following subdirectories exist:

Table 6-1 Subdirectories for ADK

Directory	Description
<root>	Contains the build procedure files.
ant	Complete ANT build system. This ANT is the standard Apache ANT build system, with some added extensions.
doc	Contains the java doc subdirectory. You may place the ADK documentation here.
doc\javadoc	The help for the ADK in javadoc format.
example	Contains our example adapter.
example\src	Contains the source java files for the example adapters.
example\deploy	Contains adapter.xml, which describes this adapter.
lib	Contains the ADK libraries needed for compilation.

An adapter is a subdirectory in the main ADK structure. After installing, **example** is one such adapter. Adapter code has to be structured in the following way

:

Table 6-2 Adapter Code Structuring Table

Directory	Description
<adapter>	The short name of the adapter. In the case of the example adapter, this is **example**.
<adapter>\src	Mandatory. The root for the source java files.
<adapter>\deploy	Mandatory. The deployment directory. This should contain a file called **adapter.xml**.
<adapter>\lib	Optional. Additional libraries to be added to the lib directory of **ISEE.war**.
<adapter>\config	Optional. Additional files to be copied to the classes directory of **ISEE.war**.

In the example provided, only **src** and **deploy** exist.

After compiling the files, create a staging directory (see the following sections for a description on how to build adapters). The staging directory can be deleted and recreated with the build procedure later.

Table 6-3 Directory Description Table

Directory	Description
staging	The root of the production directory.
staging\classes	The compiled java classes for the adapters.

Table 6-3 Directory Description Table

staging\adapters	Contains the built jars for each of the adapters. The adapters appear with the name adapter_<adaptershortname>.jar.
staging\config	The files from each config subdirectories for each adapter.
staging\deploy	The files from each of the deploy subdirectories, renamed as <adaptershortname>.xml.
staging\lib	The files from each of the lib subdirectories for each adapter.
staging\dist	The final isee.adapters deployable file.

Creating Adapter Source Structures

To create new adapters:

Step 1 Create the directory structure as defined above.

Step 2 Create the source and place it in the structure.

Step 3 Create adapter.xml and place it in the deploy directory. For more information, see the Understanding the adapter.xml Descriptor.

Step 4 Optionally add additional libraries and configuration files.

Step 5 Modify build.properties and add your adapter to the adapters line. This configures ANT to look for the created directories.

The compilation steps allow for adding the build to version control, and later compiled before installation.

Compiling Adapters

After creating the adapter, build it by executing:

build.cmd (or **./build.sh** for unix systems)

The final product appears under **staging/dist/isee.adapters**. This file needs to be provided to the administrator for deployment.

Deploying Adapters

To deploy an adapter, the administrator performs the following procedures:

Step 1 Obtain the Service Link custom adapter package. It should be in the form of a zip file.

Step 2 Unzip the adapter to a temporary directory (for example, c:\temp\adapter). This directory is hereinafter referred to as <AH>.

Step 3 Copy the <AH>/adapters/<ADAPTER_NAME>.jar to the deployed “ISEE.war/WEB-INF/lib” directory.

Step 4 Copy the <AH>/lib/* files (if any) to the deployed “ISEE.war/WEB-INF/lib” directory.

Step 5 Copy the <AH>/config/* files (if any) to the deployed “ISEE.war/WEB-INF/classes” directory.

Step 6 Copy the <AH>/udk/* files (if any) to the deployed “ISEE.war/WEB-INF/classes” directory.

Step 7 If the custom adapter is not developed internally using the Adapter Development Kit, obtain the adk.zip from the "<ServicePortal_Software_Dir>/adk" folder, where <ServicePortal_Software_Dir> is the extracted software image of the Service Catalog application. Extract the adk.zip to c:\adk (for Windows) or /opt/adk (for UNIX/Linux). This directory is hereinafter referred to as <ADK>.

Step 8 Set the JAVA_HOME environment variable if it is not already configured in the environment.

Step 9 Open a command window and cd into the <ADK>/lib folder. Execute adapter_dbinstaller.sh or adapter_dbinstaller.cmd as appropriate to your environment. Use --help or -? as the argument to the adapter installer to see the list of required input arguments. When prompted for the Adapter Deployment Descriptor file, enter the xml file name under the <AH>/deploy directory with the full path (for example, /opt/<AH>/deploy/custom_adapter.xml).

Step 10 For each udk file that was installed (Step 6), add the file’s name to the “UDConfig” property inside the integrationserver.properties file. The UDConfig property is a comma-delimited list of all udconfig files. Append the adapters udconfig files to this list.

Step 11 Start the Service Catalog and Service Link servers. Verify the new adapter exists.

Implementing an Adapter?

An adapter is the vehicle by which Service Link connects with external systems (often referred as third-party systems). Adapters are composed of three pieces:

- An inbound piece, referred to as the **inbound adapter**
- An outbound piece, referred to as the **outbound adapter**
- An error handler

The inbound adapter manages incoming communications into Service Catalog. It processes the XML messages coming into the system. There are two types of inbound adapters: pollers and listeners. A poller is a thread that periodically wakes up and looks for incoming messages, while a listener waits and is awakened by an external event. An example of a poller is the inbound file adapter, which needs to periodically check for messages. An example of a listener is the HTTP adapter which waits until an HTTP XML event is posted.

Outbound adapters manage the XML messages coming out of Service Catalog. There is only one type of outbound adapter.

An agent is a logical element that protects service designers from having to know all the complexities of adapter and connection properties. And agent defines an inbound adapter and an outbound adapter. The inbound adapter is optional and can be specified as “Auto complete”. “Auto complete” is a mode whereby the system does not need a reply from a third party for the workflow to proceed, and is mostly associated with unreliable, or shoot-and-forget protocols, such as an email-based system. The administrator configures agents and their associations with adapters for the service designers to use.

In addition, XML transformations can be applied to messages before they go to a third-party system, or after they are received from a third-party system and delivered to Service Link.

The message system uses a common XML dialect known as nsXML, which is a schema that defines the valid XML that Service Link can process or produce. nsXML currently consists of six operations:

- task-started – outgoing
- task-cancelled – outgoing
- take-action – incoming
- send-parameters – incoming
- add-comment – incoming

When outgoing, Service Link can transform these operations to a destination. The same is true for incoming messages, and the XSL transformations can convert the external format into the nsXML dialect.

For more information about nsXML operations, see Understanding Communication Message Content and Structure.

Types of Adapters

The adapters are of two types:

- Transport Adapters

  Transport adapters are specific to a given transport, such as HTTP, file, JMS, or some proprietary network socket implementation.

- Application Adapters

  Application adapters have an element of transport but are better understood by the specific third-party application, such as Remedy and Siebel. In many cases native APIs are provided through jars. In this version of Service Link, transport adapters cannot yet be extended to create application adapters.

Agents may use different adapters for inbound and outbound messages.

Adapter Components

In addition to java code, an adapter is composed of:

- Jar libraries (for example, Remedy java API)
- Static configuration files. We do not recommend changing of text files once deployed.
- Deployment descriptor. An XML file that describes the adapter.

Connection Properties

In order to connect to third-party systems, adapters may expose connection properties that the Service Link module exposes to administrators. They are described in the XML deployment descriptor, and their values can be retrieved by the java code to a well established API.

Example: Implementing a File Adapter

This section illustrates how to implement a simple adapter. The example adapter is a file adapter that communicates with the external world.

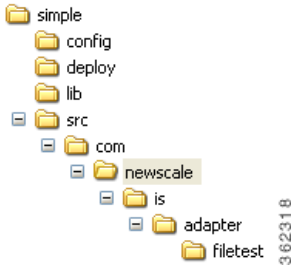
The simple file adapter contains:

- An outbound adapter that creates a file, whose file name is specified through adapter properties.
- An Inbound adapter that reads a file, whose file name is specified through adapter properties.
- A simple exception handler.

Creating Directory Structure

First, create the adapter's directory structure. In the ADK directory structure, create a directory named **simple**, and create the following directory structure under it:

Figure 6-1 Directory Structure



Under **src** notice the source package representing the java package **com.newscale.is.adapter.filetest**. Any other package can be used, but this example uses this one.

Creating Outbound Adapter Class

Secondly, create the outbound adapter class. The name is **FileOutboundAdapter** and this class file should be placed in the package described in step one. Its skeleton is shown below, without the implementation of the methods.

```
import com.newscale.is.adk.AdapterContext;
import com.newscale.is.adk.base.OutboundAdapter;
import com.newscale.is.adk.exceptions.AdapterException;
public class  FileOutboundAdapter extends OutboundAdapter {
  public FileOutboundAdapter (AdapterContext context) {
    super(context);
  }
  public void initiate (AdapterContext context) throws AdapterException {
  }
  public void processMessage (String message, String channelId)throws AdapterException {
  }
  public void terminate () throws AdapterException {
  }
  public void commit() throws AdapterException {
  }
  public void rollback() throws AdapterException {
  }
}
```

To create an outbound adapter, the class needs to extend the class **com.newscale.is.adk.base.OutboundAdapter** as shown above.

Implement a constructor that receives a **com.newscale.is.adk.AdapterContext** as a parameter. The recommended way to implement this constructor is also shown above: calling the super constructor.

Implement the initiate method as shown above. This method is called when an agent using the adapter is started. If this method is empty, you can omit it.

Implement the **processMessage** method. This method is called when a message is ready to be sent. If a transformer is specified in the agent, the transformer has transformed the message.

Implement the **terminate** method. Call this method is when the agent stops. If this method is empty, you can omit it.

Implement the **commit** method. Call this method when the agent is about to complete its transaction. If this method is empty, you can omit it. This method is used so that a transaction can be started in the **processMessage**, and later completed.

Implement the **rollback** method. This method is called when the agent is about to rollback its transaction. If this method is to be left empty, it can be omitted. This method is used so that a transaction can be started in the **processMessage**, and later recalled.

For more information about transaction support, see Configuring Transaction Notification.

In our case, the file outbound class writes a file with the contents of the xml. To achieve that, first set up a variable that keeps the file name where the file is stored. For this purpose, use the agent properties.

```
String path = null;
 public void initiate (AdapterContext context)
   throws AdapterException {
   Properties properties = context.getProperties();
   this.path = properties.getProperty("OB_FILE_DIR") + "/" +
       properties.getProperty("OB_FILE_NAME");
 }
```

When the string is received, writing it to the file is trivial.

```
  public void processMessage (String message, String channelId)
    throws AdapterException {
    try {
      Writer w = new FileWriter(path);
      w.write(message);
      w.close();
    } catch (Exception e) {
      e.printStackTrace();
      throw new AdapterException(1, "Problem while writing to a file: " +
          e.getMessage());
    }
}
```

Of course, this code has been oversimplified for the sake of explanation.

Creating Poller Inbound Adapter Class

The skeleton for our inbound adapter is as follows:

```
public class FileInboundAdapter extends InboundAdapter {
  public FileInboundAdapter (AdapterContext context) {
    super(context);
  }
  public void initiate (AdapterContext context) throws AdapterException {
  }
  public String receiveMessage () throws AdapterException {
      return null;
    }
  }
  public void terminate () throws AdapterException {
  }
  public void commit()
    throws AdapterException {
  }
```

```
  public void rollback()
    throws AdapterException {
  }
}
```

The semantics of the methods are just like the ones in the outbound adapter. The only exception is the **receiveMessage** method. The **receiveMessage** method is called periodically in the case of a poller adapter. If data is found, then the method returns a valid xml in third-party format. If no data is found, null is returned. The code for the inbound adapter is as follows (just like the outbound adapter, the initialization is done with the correct parameters):

```
String path = null;
public void initiate (AdapterContext context)
  throws AdapterException {
  Properties properties = context.getProperties();
  this.path = properties.getProperty("IB_FILE_DIR") + "/" +
      properties.getProperty("IB_FILE_NAME");
}
```

Processing of the file is done as follows:

```
public String receiveMessage () throws AdapterException {
   String receivedMessage = "";
   char data[] = {};
   try {
     StringBuffer buffer = new StringBuffer();
     FileInputStream fis = new FileInputStream(path);
     InputStreamReader isr = new InputStreamReader(fis, "UTF8");
     Reader in = new BufferedReader(isr);
     int ch;
     while ((ch = in.read()) > -1) {
       buffer.append((char) ch);
     }
     in.close();
     String requestString = buffer.toString();
     boolean success = (new File(path)).delete();
     return requestString;
   } catch (Exception e) {
     return null;
   }
}
```

Creating Listener Inbound Adapter

A listener adapter is created by virtue of an ad-hoc process. Two classes are in play: the inbound adapter class, and an actual receiver class, like a servlet. The receiver class is required to obtain the channel ID. The receiver class locates the InboundAdapter class as follows:

```
ChannelInfoVO chVo = AgentDAO.getInstance().getChannelInfo(channelId);
 if(chVo != null){
   Adapter adapter =
       AgentManager.getInstance().getAdapter(chVo.getAgentId());
   ((InboundAdapter).receiverProcess(xml);
 }
```

The inbound adapter has a method called **receiverProcess** that should be called with the message, or an object whose **toString()** method returns the text of the message. The example does not provide a listener inbound adapter.

Implementing Exception Handler

Once the two adapters are done, the exception handler needs to be implemented. In our case it is a very simple class, where all we do is output the error. The complete class is shown here:

```
public class FileExceptionHandler implements ITransportExceptionHandler {
  public FileExceptionHandler () {
  }
  public void handleException (Map props, String message) {
    System.out.println("Outbound Message Failed to deliver: " + message);
  }
}
```

Configuring Transaction Notification

Transaction support has been provided to the adapters so that agents get notified before they undo their own transactions. The methods **commit** and **rollback** have been added for that purpose.

Note No logic code should be added to these methods, as the system is in the middle of committing or rolling back a transaction. These methods should only rollback or commit their resources.

To track resources to be committed or rolled back, an adapter can use this common technique:

Create a static map. Once the processing method is called (either processMessage or receiveMessage) the method can add:

```
private static Map resources = new HashMap();
public void processMessage (String message, String channelId)
  throws AdapterException {
  Connection con = … // obtain a connection to external resource
  Map.put(Thread.getCurrentThread(), con);
}

public void commit() throws AdapterException {
  con = (Connection) map.get(Thread.getCurrentThread());
  map.remove(Thread.getCurrentThread());
  con.commit();
}
```

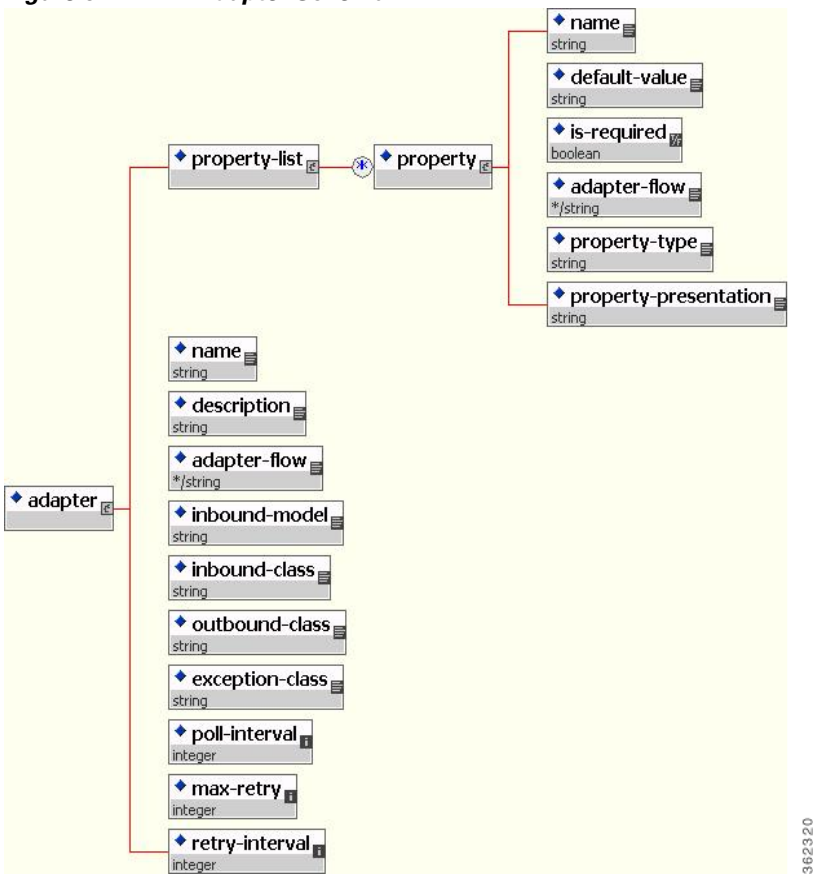
Understanding the adapter.xml Descriptor

The adapter descriptor contains information for the deployment of the adapter and its properties.

The Adapter Schema

The adapter schema is as follows:

Figure 6-2 Adapter Schema



Description of "adapter" Element Fields

name: Name of the adapter

description: Description of the adapter

adapter-flow:

Valid values for this are:

- "inbound"
- "outbound"

inbound-model:

Valid values are:

- "listener"
- "poller"
- "extendedpoller"

inbound-class: Absolute class name of inbound adapter

outbound-class: Absolute class name of outbound adapter

exception-class: Absolute class name of exception handler for this adapter

poll-interval: Poll interval (applicable for “poller” type adapter) in milliseconds

max-retry: Max number of retries in case of message failure

retry-interval: Interval between retries in milliseconds

Description of “property” (Adapter Properties) Element Fields

name: Name of the adapter property

default-value: Default value for the property

is-required: Whether this is a mandatory or optional property. Valid values are “true” or “false”

property-type: The type of property. Valid values are “string” for now.

property-presentation: Valid values are “text” and “password”

adapter-flow:

Valid values are:

- “inbound”
- “outbound”

Adapter.xml Example

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter>
<property-list>
<property>
<name>InboundFinalResolution</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OnError</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
```

```
<name>BackupSuffix</name>
<default-value>.bak</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>FileNameDateFormat</name>
<default-value>.yyyyMMddHHmmssSSS</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundConflictResolution</name>
<default-value>Rename</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
</property-list>
<name>File Adapter</name>
<description>Read/write the external data from/to external file system</description>
<adapter-flow>inbound</adapter-flow>
<inbound-model>poller</inbound-model>
<inbound-class>com.newscale.is.adapter.file.FileInboundAdapter</inbound-class>
<outbound-class>com.newscale.is.adapter.file.FileOutboundAdapter</outbound-class>
<exception-class>com.newscale.is.adapter.file.FileExceptionHandler</exception-class>
<poll-interval>10000</poll-interval>
```

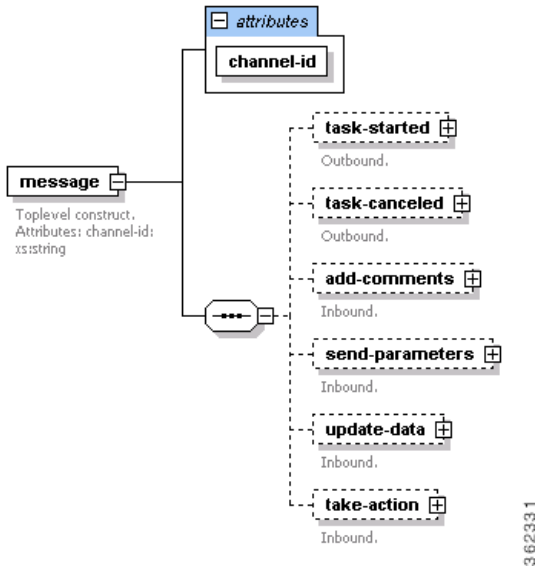
```
<max-retry>0</max-retry>
<retry-interval>0</retry-interval>
</adapter>
```

Understanding Communication Message Content and Structure

This section describes the details of the communication message content and structure. The message content is encapsulated in XML documents which are sent between Service Catalog and third-party systems over various carrier protocols such as HTTP, SOAP, or JMS. For easier understanding of the structures and substructures of messages, a graphical notation is used.

Message

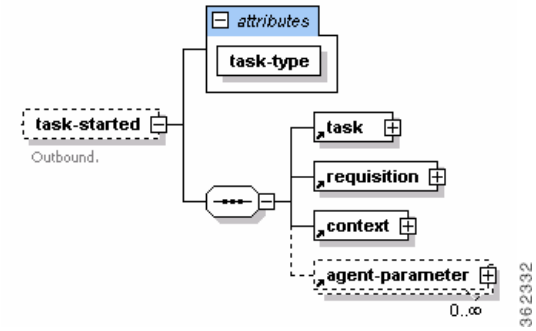
Figure 6-3 Message



The outbound \has a top level element message which contains the element “task-started” or “task-canceled”. The inbound document has a top level element message which contains one or many elements “add-comments,” “send-parameters,” or “take-action”. The message tag has a mandatory attribute which is called channel-id and is of type string. It is a unique string value created by Service Link for any outbound message created. The third-party system needs to reply back the message with the corresponding channel-id. This ID has to be carried on both the Service Catalog and third-party system sides.

Task Started or Task Cancelled

Figure 6-4 Task Started or Task Cancelled



Task started kicks off an external activity in the third-party system. The design strategy followed for this operation is to incorporate all the information that may be required by the third-party system to execute the task. This element holds all the required details about the task and the requisition it belongs to. It may also contain one or more optional agent parameters. The context element describes the task in context of service delivery plan. This node does not contain values for Requisition-level reviews and approvals.

Task

Figure 6-5 Task

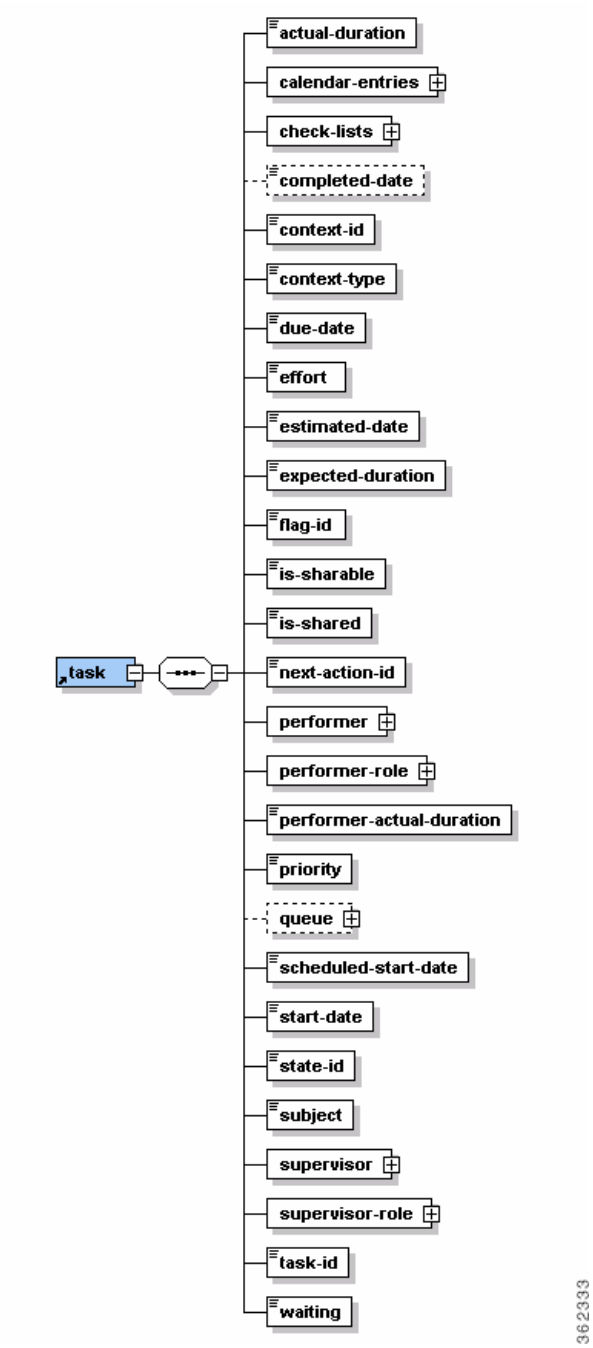
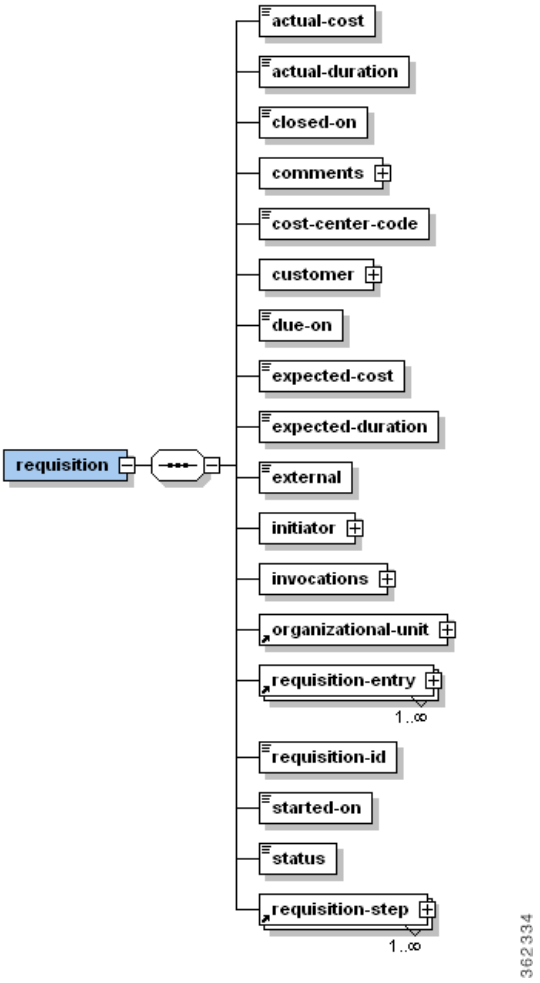


Table 6-4 Task Element and Description

Element	Description
actual-duration	Empty because the task has just become ongoing.
calendar-entries	The calendar entry of the person who requested a service.
check-lists	Task check list.
completed-date	Empty because the task has not yet been completed.
context-id	The object id of the context object in which the task gets initiated.
context-type	The object context, for example, Requisition Entry.
due-date	The due date for the task.
effort	Expected task effort in hours (how many working hours are expected to be required by one person to complete the task?).
estimated-date	Estimated completion date and time of the task.
expected-duration	Expected task duration in hours (how many working hours are expected to pass from the beginning of the task to the end?)/.
flag-id	Color indicator for the UI.
is-sharable	Boolean value indicating whether the task is sharable or not.
is-shared	Boolean value indicating whether the task is shared or not.
next-action-id	What is the next possible action for the task?
performer	The performer of the task. The performer element has an associated person object/.
performer-actual-duration	Empty because the task has not yet been completed/.
performer-role	What is the process role the performer is fulfilling?
priority	Task priority: 1, 2 or 3 for high, medium, or low, respectively/.
queue	Description of the queue to which this task has been assigned.
scheduled-start-date	Date on which the task is scheduled to be started/.
start-date	Date on which the task was started/.
state-id	Which state the task is in/.
subject	Subject of the task.The subject changes with the service definition, not with the requisition entry/.
supervisor	The supervisor of the task.The supervisor element has an associated person object.
supervisor-role	The process role the supervisor is fulfilling.
task-id	An integer used to uniquely identify this task instance. A new number is generated for each task of a requisition entry.
waiting	An indicator that represents the dependencies of this task including sub tasks and third-party systems.

Requisition

Figure 6-6 Requisition



The requisition element encapsulates all requisition and requisition entry data that can be used for integration purposes when executing an external activity.

Table 6-5 Requisition Element Description Table

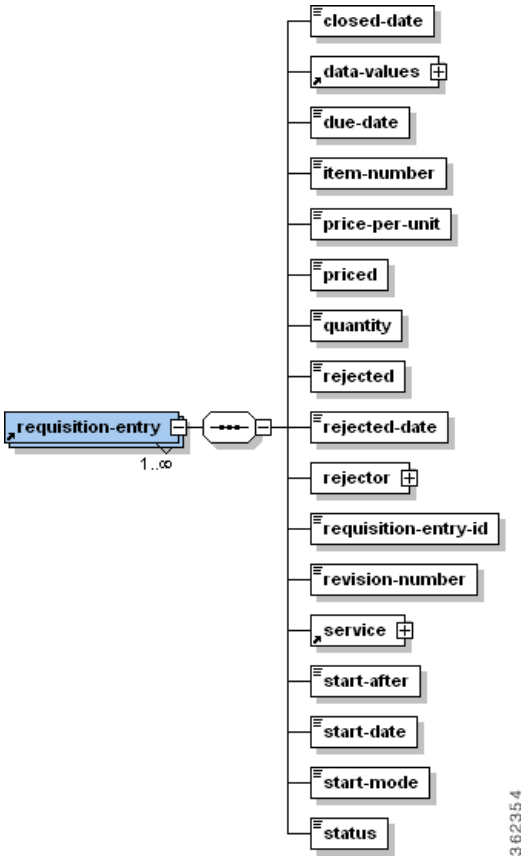
Element	Description
services	Number of services (or requisition entries) requested.
actual-cost	Actual cost of the requisition.
actual-duration	Actual duration of the requisition.
closed-on	Empty, as the requisition has not yet been completed.
comments	Comment on the requisition.
cost-center-code	Not used.
customer	The person for whom the requisition was ordered. It holds the person object data.
due-on	Date and time when the delivery of the requisition is due.
expected-cost	Expected cost of the requisition.

Table 6-5 Requisition Element Description Table

Element	Description
expected-duration	Expected duration in hours for handling the whole requisition.
external	Boolean value indicating whether the requisition has been initiated from an external system.
initiator	The person who ordered this requisition. It holds the person object data.
invocations	Attributes set up through RAPI (Requisition API).
organizational-unit	The organizational unit of the requestor (initiator).
requisition-entry	The requisition entry data.
requisition-id	Integer id of the submitted requisition. This is the same ID that can be seen in My Services and Service Manager after submitting a requisition manually.
requisition-step	The requisition authorization/delivery steps and status.
started-on	The date on which the requisition started.
status	State the requisition is currently in. While executing an external activity, this is ongoing.

Requisition Entry

Figure 6-7 Requisition Entry



This tag encapsulates all the data of one requisition entry that can be used for integration purposes.

Table 6-6 Requisition Entry Element Description Table

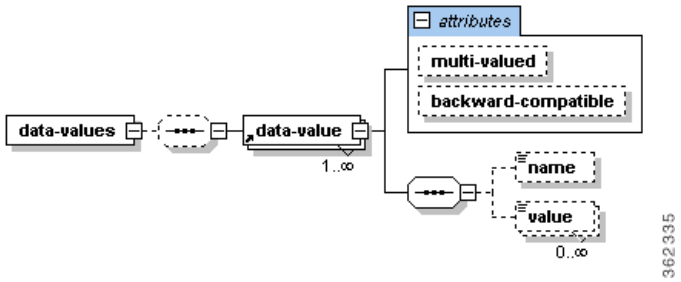
Element	Description
closed-date	Date and time when the requisition entry's status was changed from ongoing to completed. It is empty because the requisition is not closed when the task is ongoing.
data-values	Requisition entry data value.
due-date	Date on which this requisition is supposed to finish.
item-number	Item number of the requisition entry within the requisition.
price-per-unit	Unit price of the service requested.
priced	True if the price has been established and false if pricing is not done on the requisition.
quantity	Quantity ordered.
rejected	Indicates whether the requisition entry is approved or rejected.
rejected-date	If it is rejected, on what date.
rejector	Indicates the person who rejected the requisition.

Table 6-6 Requisition Entry Element Description Table

requisition-entry-id	Entry ID.
revision-number	If the revision is made it indicates the revision number.
service	Element related to the service which this requisition entry belongs to.
start-after	Delayed start date.
start-date	The date on which the requisition entry got started.
start-mode	Specifies if the requisition entry starts immediately or late.
status	Status of the requisition entry: closed or ongoing.

Data Values

Figure 6-8 Data Values



The data-values element can have one or more data values comprised of dictionary data. The data-value name indicates the “Dictionaryname.FieldName” and value is the value entered by the user while ordering the service. If the value is a multi-select drop down list, then one data-value element can have multiple values.

Service

Figure 6-9 Service

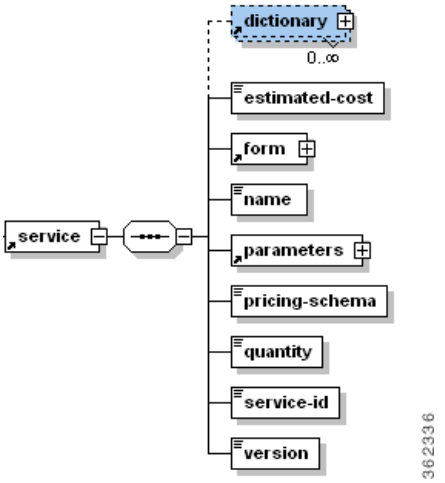


Table 6-7 Service Element Description Table

Element	Description
dictionary	A service element can have zero or more dictionaries.
estimated-cost	The estimated cost of the service.
form	Element which holds all the field elements of the service form.
name	Name of the service.
parameters	Parameters defined for this service.
pricing-schema	Specifies if the service is a bid, pricing task or fixed price.
quantity	How many quantities were ordered?
service-id	Id of the service in Service Catalog.
version	Last modified version number of the service.
standard-duration	Standard duration for the service.

Dictionary

Figure 6-10 Dictionary

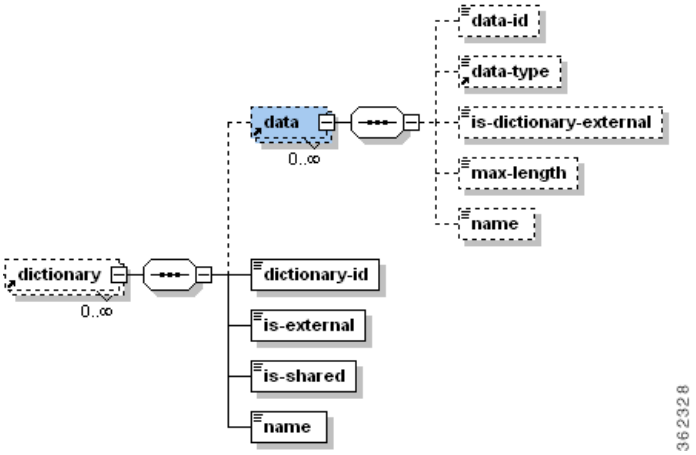


Table 6-8 Dictionary Element Description Table

Element	Description
caption	A string value containing the caption data within the dictionary.
data	The data elements within the dictionary. The data element holds values for the data element name, maximum length, data type and other metadata.
dictionary-id	The dictionary id of a dictionary within Service Catalog.
dictionary-template-type-id	The template used for creating the dictionary (for example, 2 for person-based dictionaries).
classification-id	The dictionary classification (applicable to Service Item dictionaries only).
mdr-data-type-id	The dictionary service item type (applicable to Service Item dictionaries only).
display-order	An integer value containing the display order of the dictionary.
is-external	A Boolean value which indicates whether the dictionary is an internal Service Catalog dictionary or is external.
is-reportable	A Boolean value stating whether the dictionary has been marked as reportable for use with the Advanced Reporting module's Ad-Hoc reporting feature.
is-shared	A Boolean value which indicates whether the dictionary is a shared dictionary or not.
is-template	A Boolean value which indicates whether the dictionary is a template; the value is always false.
logic-name	Internal name of the dictionary (applicable to reserved dictionaries only).
name	Name of the dictionary.

Form

Figure 6-11 Form

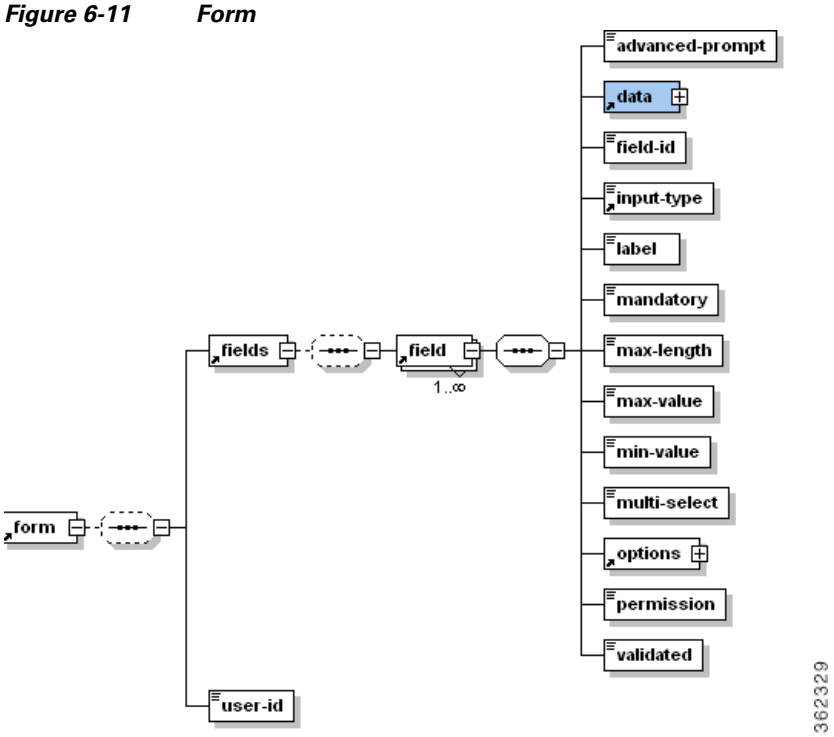
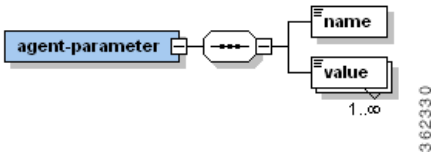


Table 6-9 Form Element Description Table

Element	Description
fields	Fields have one or many field elements inside a requisition form. advanced-prompt – Rich html prompt. data – holds the data for the field which has data type, name, length, and so on. dictionary-display-order – The value for dictionary-display-order is the value of DefObjectDictionaries.DisplayOrder for the Dictionary associated with the DataElement associated with the Field. display-order – the value for **display-order** is the value of DefObjectDataHTML.DisplayOrder for the field. field-id – Field Id within the Service Catalog database. input-type – Input type of the field (for example, text, option, and so on). label – Label specified for the field. mandatory – The field data is mandatory in the service. max-length – Maximum length specified for the field. max-value – If it is a number range specified. min-value multi-select – Whether the input type is a multi-select box. options – The option list available for this data field. permission – validated – Should it be validated or not.
user-id	

Agent Parameter

Figure 6-12 Agent Parameter



The agent parameter represents the external parameters specified for the agent. It has the Boolean attribute called multi-valued which is either true or false based on whether this parameter has multiple values selected by user. The name represents the name of the agent parameter and value represents its value.

Examples: Inbound and Outbound Documents

task-started or task-canceled (outgoing)

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="18071221:1124919814742:-32752"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <task-started task-type="task">
        <task>
            <actual-duration>0.0</actual-duration>
            <calendar-entries>
                <calendar-entry>
                    <calendar-entry-id>2</calendar-entry-id>
                    <date>Thu Aug 25 17:00:00 PDT 2005</date>
                    <end-time>Fri Aug 26 21:40:37 PDT 2005</end-time>
                    <is-blocked>false</is-blocked>
                    <is-break>false</is-break>
                    <is-read>false</is-read>
                    <person>
                        <company-address/>
                        <email>admin@company.com</email>
                        <fax/>
                        <first-name>admin</first-name>
                        <home-ou>
                            <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                            <organizational-unit-id>1</organizational-unit-id>
                        </home-ou>
                        <home-phone/>
                        <last-name/>
                        <login-name>admin</login-name>
                        <person-id>1</person-id>
                        <personal-address/>
                        <supervisor-name/>
                        <timezone>Pacific Standard Time</timezone>
                        <work-phone/>
                    </person>
                    <sequence>0</sequence>
                    <start-time>Thu Aug 25 21:40:37 PDT 2005</start-time>
                    <subject>External Task</subject>
                </calendar-entry>
            </calendar-entries>
            <check-lists>
                <check-list-entry>
                    <display-order>1</display-order>
                    <is-mandatory>true</is-mandatory>
                    <last-date/>
                    <last-person/>
                    <name>Make sure you wake up</name>
                    <status>false</status>
                </check-list-entry>
                <check-list-entry>
                    <display-order>2</display-order>
                    <is-mandatory>true</is-mandatory>
                    <last-date/>
                    <last-person/>
                    <name>Make sure you take a shower</name>
                    <status>false</status>
                </check-list-entry>
                <check-list-entry>
                    <display-order>3</display-order>
```

```
                        <is-mandatory>true</is-mandatory>
                        <last-date/>
                        <last-person/>
                        <name>Make sure you have breakfast</name>
                        <status>false</status>
                    </check-list-entry>
                </check-lists>
                <completed-date/>
                <context-id>1</context-id>
                <context-type>Requisition Entry</context-type>
                <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
                <effort>10.0</effort>
                <estimated-date/>
                <expected-duration>10.0</expected-duration>
                <flag-id>0</flag-id>
                <is-sharable>false</is-sharable>
                <is-shared>false</is-shared>
                <next-action-id>2</next-action-id>
                <performer>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name/>
                    <login-name>admin</login-name>
                    <person-id>1</person-id>
                    <personal-address/>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </performer>
                <performer-actual-duration>0.0</performer-actual-duration>
                <priority>2</priority>
                <scheduled-start-date>Thu Aug 25 21:40:37 PDT 2005</scheduled-start-date>
                <start-date>Wed Aug 24 21:42:15 PDT 2005</start-date>
                <state-id>2</state-id>
                <subject>External Task</subject>
                <supervisor>
                    <company-address>Foo Bar 25 Suite 300 Foo City CA  94404
USA</company-address>
                    <email>internal@company.com</email>
                    <fax/>
                    <first-name>Monkey</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name>McBride</last-name>
                    <login-name>monkey</login-name>
                    <person-id>3</person-id>
                    <personal-address>Fuchi Caca 16 Apartment C Fuchi Minn OR  78787
USA</personal-address>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </supervisor>
                <task-id>3</task-id>
                <waiting>1</waiting>
```

```
            </task>
            <requisition>
                <actual-cost>0.0</actual-cost>
                <actual-duration>0.0</actual-duration>
                <closed-on/>
                <comments>
                    <comment>
                        <comment-date>Wed Aug 24 21:42:06 PDT 2005</comment-date>
                        <comment-id>1</comment-id>
                        <comment-text>I am adding a comment and I cannot think of a better
comment</comment-text>
                        <component-id>3</component-id>
                        <component-name>Request Center Component</component-name>
                        <person>
                            <company-address/>
                            <email>admin@company.com</email>
                            <fax/>
                            <first-name>admin</first-name>
                            <home-ou>
                                <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                                <organizational-unit-id>1</organizational-unit-id>
                            </home-ou>
                            <home-phone/>
                            <last-name/>
                            <login-name>admin</login-name>
                            <person-id>1</person-id>
                            <personal-address/>
                            <supervisor-name/>
                            <timezone>Pacific Standard Time</timezone>
                            <work-phone/>
                        </person>
                        <source-object-id>2</source-object-id>
                        <source-object-inst-id>1</source-object-inst-id>
                    </comment>
                </comments>
                <cost-center-code/>
                <customer>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
                        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
                        <organizational-unit-id>1</organizational-unit-id>
                    </home-ou>
                    <home-phone/>
                    <last-name/>
                    <login-name>admin</login-name>
                    <person-id>1</person-id>
                    <personal-address/>
                    <supervisor-name/>
                    <timezone>Pacific Standard Time</timezone>
                    <work-phone/>
                </customer>
                <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
                <expected-cost>0.0</expected-cost>
                <expected-duration>0.0</expected-duration>
                <external>false</external>
                <initiator>
                    <company-address/>
                    <email>admin@company.com</email>
                    <fax/>
                    <first-name>admin</first-name>
                    <home-ou>
```

```
            <name>&lt;s ID=&quot;847&quot;/&gt;</name>
            <organizational-unit-id>1</organizational-unit-id>
        </home-ou>
        <home-phone/>
        <last-name/>
        <login-name>admin</login-name>
        <person-id>1</person-id>
        <personal-address/>
        <supervisor-name/>
        <timezone>Pacific Standard Time</timezone>
        <work-phone/>
    </initiator>
    <invocations/>
    <organizational-unit>
        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
        <organizational-unit-id>1</organizational-unit-id>
    </organizational-unit>
    <requisition-entry>
        <closed-date/>
        <data-values>
            <data-value>
                <name>Requester</name>
                <value>John McGarzafi</value>
            </data-value>
            <data-value>
                <name>RemedyStuff.TicketID</name>
                <value>None yet</value>
            </data-value>
            <data-value>
                <name>RemedyStuff.AssetNumber</name>
                <value>123456789</value>
            </data-value>
        </data-values>
        <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
        <item-number>1</item-number>
        <price-per-unit>0.0</price-per-unit>
        <priced>true</priced>
        <quantity>1</quantity>
        <rejected>false</rejected>
        <rejected-date/>
        <rejector>
            <company-address/>
            <email/>
            <fax/>
            <first-name/>
            <home-phone/>
            <last-name/>
            <login-name/>
            <person-id>0</person-id>
            <personal-address/>
            <supervisor-name/>
            <timezone/>
            <work-phone/>
        </rejector>
        <requisition-entry-id>1</requisition-entry-id>
        <revision-number>5</revision-number>
        <service>
            <dictionary>
                <data>
                    <data-id>3</data-id>
                    <data-type>Person</data-type>
                    <is-dictionary-external>false</is-dictionary-external>
                    <max-length>100</max-length>
                    <name>Requester</name>
```

```
    </data>
    <dictionary-id>1</dictionary-id>
    <is-external>false</is-external>
    <is-shared>false</is-shared>
    <name>Monkey Service (private)</name>
</dictionary>
<dictionary>
    <data>
        <data-id>1</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>TicketID</name>
    </data>
    <data>
        <data-id>2</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>AssetNumber</name>
    </data>
    <dictionary-id>2</dictionary-id>
    <is-external>false</is-external>
    <is-shared>true</is-shared>
    <name>RemedyStuff</name>
</dictionary>
<estimated-cost>0.0</estimated-cost>
<form>
    <fields>
        <field>
            <advanced-prompt/>
            <data>
                <data-id>2</data-id>
                <data-type>Text</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>50</max-length>
                <name>AssetNumber</name>
            </data>
            <field-id>2</field-id>
            <input-type>text</input-type>
            <label>AssetNumber</label>
            <mandatory>false</mandatory>
            <max-length>50</max-length>
            <max-value>0.0</max-value>
            <min-value>0.0</min-value>
            <multi-select>false</multi-select>
            <options>
                <available-keys/>
                <available-labels/>
                <current-values/>
                <multivalued>false</multivalued>
            </options>
            <permission>4</permission>
            <validated>true</validated>
        </field>
        <field>
            <advanced-prompt/>
            <data>
                <data-id>1</data-id>
                <data-type>Text</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>50</max-length>
                <name>TicketID</name>
            </data>
```

```
                        <field-id>1</field-id>
                        <input-type>text</input-type>
                        <label>TicketID</label>
                        <mandatory>false</mandatory>
                        <max-length>50</max-length>
                        <max-value>0.0</max-value>
                        <min-value>0.0</min-value>
                        <multi-select>false</multi-select>
                        <options>
                            <available-keys/>
                            <available-labels/>
                            <current-values/>
                            <multivalued>false</multivalued>
                        </options>
                        <permission>4</permission>
                        <validated>true</validated>
                    </field>
                    <field>
                        <advanced-prompt>Give the name!</advanced-prompt>
                        <data>
                            <data-id>3</data-id>
                            <data-type>Person</data-type>
                            <is-dictionary-external>false</is-dictionary-external>
                            <max-length>100</max-length>
                            <name>Requester</name>
                        </data>
                        <field-id>3</field-id>
                        <input-type>text</input-type>
                        <label>Requester Name</label>
                        <mandatory>false</mandatory>
                        <max-length>100</max-length>
                        <max-value>0.0</max-value>
                        <min-value>0.0</min-value>
                        <multi-select>false</multi-select>
                        <options>
                            <available-keys/>
                            <available-labels/>
                            <current-values>
                                <string>John McGarzafi</string>
                            </current-values>
                            <multivalued>false</multivalued>
                        </options>
                        <permission>4</permission>
                        <validated>true</validated>
                    </field>
                </fields>
                <user-id>0</user-id>
            </form>
            <name>Monkey Service</name>
            <parameters>
                <default-duration>0.0</default-duration>
                <priority>2</priority>
                <start-date/>
                <start-mode>0</start-mode>
            </parameters>
            <pricing-schema>0</pricing-schema>
            <quantity>1</quantity>
            <service-id>1</service-id>
            <version>5</version>
        </service>
        <start-after/>
        <start-date>Wed Aug 24 21:40:50 PDT 2005</start-date>
        <start-mode>0</start-mode>
        <status>1</status>
```

```
            </requisition-entry>
            <requisition-id>1</requisition-id>
            <started-on>Wed Aug 24 21:40:32 PDT 2005</started-on>
            <status>1</status>
            <requisition-step>
                <completed-on/>
                <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
                <estimated-on>Fri Aug 26 21:40:37 PDT 2005</estimated-on>
                <name>Delivery project for Monkey Service</name>
                <status>2</status>
            </requisition-step>
        </requisition>
        <agent-parameter multi-valued="false">
            <name>Ticket</name>
            <value>None yet</value>
        </agent-parameter>
        <agent-parameter multi-valued="false">
            <name>Asset</name>
            <value>123456789</value>
        </agent-parameter>
    </task-started>
</message>
```

take-action (incoming)

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="18071221:1124919814742:-32752">
            <take-action action="done"/>
</message>
```

send-parameters (incoming)

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="18071221:1116468068789:-32360">
    <send-parameters>
      <agent-parameter>
         <name>Param1</name>
             <value>cat</value>
      </agent-parameter>
      <agent-parameter>
         <name>Param2</name>
             <value>catlitter</value>
       </agent-parameter>
     </send-parameters>
</message>
```

add-comments (incoming)

```
<?xml version="1.0" encoding="UTF-8"?>
<message  channel-id="32580443:1116276793649:-32629">
            <add-comments>
               <comment>Test Comment</comment>
            </add-comments>
</message>
```