Gentran Integration Suite™

Service Developer's Guide

Version 4.0

Sterling Commerce
An IBM Company

February 2005

STERLING COMMERCE SOFTWARE

*****TRADE SECRET NOTICE*****

THIRD PARTY SOFTWARE

Sun, Sun Microsystems, the Sun Logo, Java, JDK, the Java Coffee Cup logo, JavaBeans, JDBC, JMX and all JMX based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. All other trademarks and logos are trademarks of their respective owners.

DTD GENERATOR SOFTWARE AND POCKETSOAP SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the DTD Generator Software (Copyright © 2001 Michael H. Kay.) ("DTD Generator Software"), and the PocketSoap Software (Copyright © 2000-2003 Simon Fell) ("PocketSoap Software"). The DTD Generator Software and the PocketSoap Software are independent from and not linked or compiled with the Sterling Commerce Software. The DTD Generator Software and PocketSoap Software are free software products which can be distributed and/or modified under the terms of the Mozilla Public License version 1.0 (DTD Generator Software) and the Mozilla Public Licensed version 1.1 (PocketSoap Software) as published by The Mozilla Organization.

A copy of the Mozilla Public Licenses are provided at installdir/dtd_generator/License.txt, installdir/dtd_generator/Mozilla_Exhibit_A.txt, C:/Program Files/Sterling Commerce/Map Editor/3rd_Party/PocketSoapLicense.txt, and C:/Program Files/Sterling Commerce/Map Editor/3rd_Party/Mozilla_Exhibit_A.txt.

This license only applies to the DTD Generator Software and the PocketSoap Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

The DTD Generator Software and the PocketSoap Software are each distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the license for the specific language governing rights and limitations under the license. Original Code for the DTD Generator Software is DTD Generator 7.0 and the Initial Developer of the Original Code is Michael H. Kay. Portions created by Michael H. Kay are Copyright (C) 2001 Michael H. Kay. All Rights Reserved.

Original Code for the PocketSoap Software is PocketSOAP v1.4.3 and the Initial Developer of the Original Code is Simon Fell. Portions created by Simon Fell are Copyright © 2000-2003 Simon Fell. All Rights Reserved. Contributor David Buksbaum.

Sterling Commerce has not made any modifications to the DTD Generator Software, or the PocketSoap Software. Source code for the DTD Generator Software is located at http://saxon.sourceforge.net/dtdgen.html. Source code for the PocketSoap Software is located at http://www.pocketsoap.com/pocketsoap/.

ICE SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the ICE Software (Copyright © 1997 ICE Engineering, Inc./Timothy Gerard Endres.) ("ICE Software"). The ICE Software is independent from and not linked or compiled with the Sterling Commerce Software. The ICE Software is free software which can be distributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License or any later version.

A copy of the GNU General Public License is provided at installdir/jar/jniregistry/1_2/ICE_License.txt. This license only applies to the ICE Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

The ICE Software was modified slightly in order to fix a problem discovered by Sterling Commerce involving the RegistryKey class in the RegistryKey.java in the JNIRegistry.jar. The class was modified to comment out the finalize () method and rebuild of the JNIRegistry.jar file.

Source code for the bug fix completed by Sterling Commerce on January 8, 2003 is located at: installdir/jar/jniregistry/1_2/RegistryKey.java. Source code for all other components of the ICE Software is located at http://www.trustice.com/java/jnireg/index.shtml.

MYSQL SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the MySQL Software (Copyright © 1995-2002 MySQL AB) ("MySQL Software"). Before installing the MySQL Software, the terms and conditions of the MySQL license must be accepted.

A copy of the MySQL license is provided at installdir/mysql/MySQL_License.txt. This license only applies to the MySQL Software and does not apply to the Sterling Commerce Software, or any other Third Party Licensor Software.

JBOSS SOFTWARE, JFREE CHART SOFTWARE, JEXCEL SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the JBoss Software (Copyright © 1999-2002 JBoss.org) ("JBoss Software"), the JFree Chart Software (Copyright © 2000-2003, by Simba Management Limited and Contributors.) ("JFree Software"), and The Java Excel API (Copyright © 2002 Andrew Khan) ("JExcel Software"). The JBoss Software, the JFree Software, and the JExcel Software are independent from and not linked or compiled with the Sterling Commerce Software. The JBoss Software, the JFree Software, and the JExcel Software are free software products which can be distributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License or any later version.

A copy of the GNU Lesser General Public License is provided at installdir/noapp/lib/JBoss_License.txt, installdir/jar/jfreechart/0_9_18/JFreeChart_License.txt, and installdir/jar/jexcelapi/2_4_4/jexcel_license.txt. This license only applies to the JBoss Software, the JFree Software, and the JExcel Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

The JBoss Software was modified slightly in order to allow the ClientSocketFactory to return a socket connected to a specific host in order to control the host interfaces regardless of whether the ClientSocket Factory specified was custom or not. Changes were made to org.jnp.server.Main. Details concerning this change can be found at http://sourceforge.net/tracker/?func=detail&aid=1008902&group_id=22866&atid=376687.

Source code for the modifications completed by Sterling Commerce on August 13, 2004 is located at http://sourceforge.net/tracker/?func=detail&aid=1008902&group_id=22866&atid=376687. Source code for all other components of the JBoss Software is located at http://www.jboss.org.

PJA SOFTWARE and TEE UTILITY SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as the PJA Toolkit (Copyright 1997-2004 eTeks) (“PJA Software”) and the Tee Utility Software (Copyright © 2002 Karl M. Syring) (“Tee Utility Software”). The PJA Software and the Tee Utility Software are independent from and not linked or compiled with the Sterling Commerce Software. The PJA Software and the Tee Utility Software are free software products which can be distributed and/or modified under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License or any later version.

A copy of the GNU General Public License is provided at installdir/jar/pja/2_4/PJA_License.txt and at installdir/Tee_Utility.txt. This license only applies to the PJA Software and the Tee Utility Software and does not apply to the Sterling Commerce Software, or any other Third Party Software.

COMMONS FILE UPLOAD SOFTWARE, REG EXP SOFTWARE, ANT SOFTWARE, JETSPEED SOFTWARE, TAGLIBS SOFTWARE, COMMONS COMPONENTS SOFTWARE

The Sterling Commerce Software is distributed on the same storage media as certain software provided by the Apache Software Foundation, which may be subject to Apache Software License 2.0, such as Apache Jakarta Commons File Upload Package (“Commons File Upload Software”), Apache Regular Expression Package (“Reg Exp Software”), Apache Ant versions 1.6.1 and higher (“Ant Software”), Apache Jetspeed (“Jetspeed Software”), Apache Jakarta Xtags Library (“Taglibs Software”), and Apache Jakarta Commons Components (“Commons Components Software”). Sterling Commerce Software is neither a Derivative Work nor a Contribution as defined in the Apache Software License 2.0.

A copy of the Apache 2.0 license is provided at installdir/jar/commons_upload/1_0/ CommonsFileUpload_License.txt, installdir/jar/jetspeed/1_4/ RegExp_License.txt, installdir/jar/jetspeed/1_4/Jetspeed_License.txt, installdir/ant/Ant_License.txt, installdir/jar/xtags/1_0/Xtags_License.txt, installdir/jar/commons_httpclient/2_0_0/Commons_License.txt and at installdir/jar/commons_logging/1_0_3/ Commons_License.txt. The Apache 2.0 License applies to the Commons File Upload Software, Jetspeed Software, the Taglibs Software and the Commons Components Software, may apply to the Reg Exp Software, and Ant Software, but does not apply to the Sterling Commerce Software, or any other Third Party Licensor Software.

WARRANTY DISCLAIMER

This documentation and the Sterling Commerce Software which it describes are licensed either "AS IS" or with a limited warranty, as set forth in the Sterling Commerce license agreement. Other than any limited warranties provided, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE. The applicable Sterling Commerce entity reserves the right to revise this publication from time to time and to make changes in the content hereof without the obligation to notify any person or entity of such revisions or changes.

Sterling Commerce does not warrant or represent that use of the Sterling Commerce Software will insure compliance with the Health Insurance Portability and Accountability Act (HIPAA) Transaction Code Set Rules, and product users should consult independent legal counsel and technical support to insure compliance with HIPAA Transaction Code Set Rules and other legal requirements.

The Third Party Software is provided “AS IS” WITHOUT ANY WARRANTY AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. FURTHER, IF YOU ARE LOCATED OR ACCESSING THIS SOFTWARE IN THE UNITED STATES, ANY EXPRESS OR IMPLIED WARRANTY REGARDING TITLE OR NON-INFRINGEMENT ARE DISCLAIMED.

Without limiting the foregoing, the ICE Software, JBoss Software, JFree Software, JExcel Software, PJA Software, and Tee Utility Software are all distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Sterling Commerce, Inc.

4600 Lakehurst Court Dublin, OH 43016-2000 * 614/793-7000

Contents

Chapter 1

Gentran Integration Suite Architecture

The purpose of the Gentran Integration Suite *Service Developer's Guide* is to provide developers with the information they need to develop a service or adapter. This guide includes an overview of the Gentran Integration Suite framework and sample code to expedite the development process.

Note: Because all adapters are a type of service, this guide uses the term *service* for both services and adapters. This guide uses the term *adapter* when the information is unique to adapters. For more information about adapters, see *Adapters* on page 4.

This section covers the following topics:

- About Gentran Integration Suite
- Components of a Service
- Relationship Between Business Processes and Services
- GIS Components and a J2EE Environment

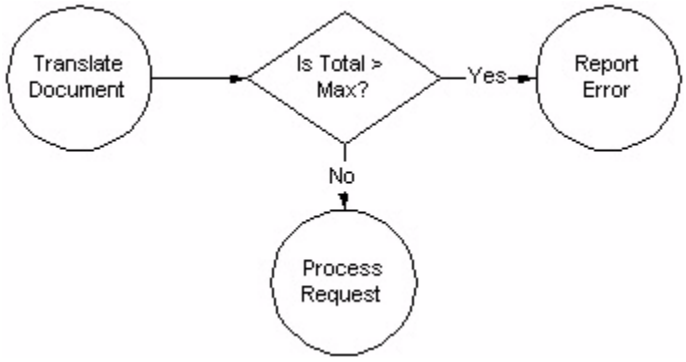
About Gentran Integration Suite

Gentran Integration Suite executes customer-specific business processes. An XML-based business process model directs the order of all processing activities in Gentran Integration Suite. This process model makes Gentran Integration Suite adaptable to a variety of processing situations.

Business Process Definitions

Gentran Integration Suite business process definitions are based on the draft Business Process Modeling Language (BPML) specification from the Business Process Management Initiative (www.bpmi.org). Because business process definitions are stored in XML, a business analyst can define business processes in several ways. An analyst can create definitions using a graphical editor, simple text editor, or any graphical process editor that can export the XML format recognized by Gentran Integration Suite.

The following figure shows a business process. A circle represents activities and a diamond represents a decision point.



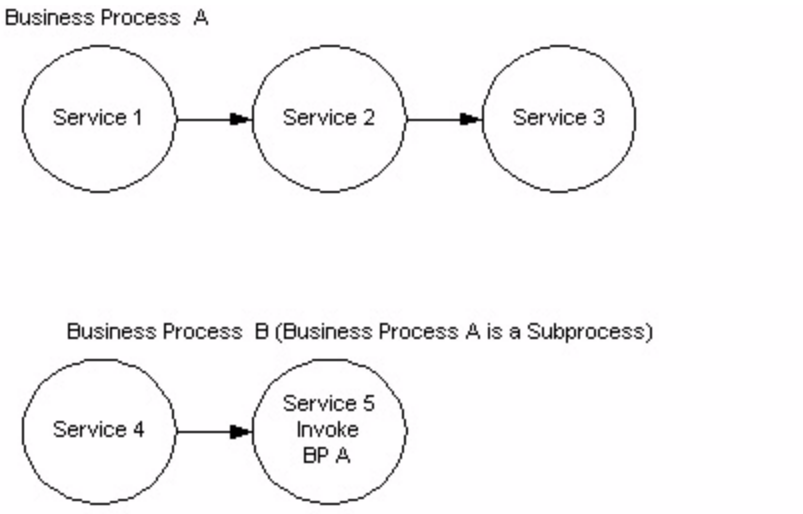
Services

Gentran Integration Suite views every activity in a business process as a *service*. A service can initiate:

- Legacy programs
- ERP systems
- Perl scripts
- Java code
- Decision engines
- Any computer program

A service in a business process can also invoke another business process, making that business process a subprocess. If the subprocess changes, the changes are reflected in all business processes that include that subprocess.

The following figure shows a subprocess:



Gentran Integration Suite supports reuse of business processes. Reuse enables you to determine what should be implemented as a service, a business process, or subprocess.

Reuse also enables a business analyst to work with information technology staff to determine whether the business processes should be written with multiple reusable components or as a single large service. For example, RosettaNet™ support can be implemented as multiple activities strung together to form a business process or as a single service.

Types of Services

There are several basic types of services in Gentran Integration Suite. The following table describes the various service types:

Type	Description
Internal	Services that are completely inside Gentran Integration Suite. Although internal services accept parameters and produce results, they do not directly interact with external systems (systems outside Gentran Integration Suite).
Input	Services that must receive data from external systems.
Output	Services that must send data to external systems.
Transport Adapter	Services that use communications protocols like FTP and HTTP to bring data into Gentran Integration Suite.
Application Adapter	Services that interact with external application systems.

Adapters

Adapters are services that interact with external systems.

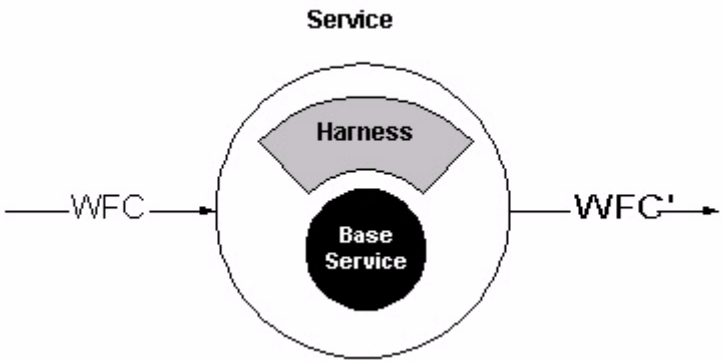
For more information about starting a business process, see *Starting a Business Process* on page 6.

Components of a Service

Every service accepts a business process state and produces a modified business process state or workflow context (WFC). Every service also has a harness. For the service, the harness performs the following functions:

1. Receives the input WFC
2. Extracts the information from it that the service needs
3. Runs the service
4. Places the results from the service in a new WFC or output for future steps in the business process workflow

The following figure shows a business process:



Example of a Service

To call an inventory system, the harness extracts the product ID from a document, sets up the calling parameters, sends the request to the external inventory system, extracts the inventory status from the return, and puts the appropriate status code in the workflow context.

Workflow Context

The *workflow context (WFC)* represents the business process state after each service has run. The WFC input to a service is written to a database. The service is complete after the new WFC is placed in persistent storage. Thus, the collection of persisted WFCs represents the state of all business processes in Gentran Integration Suite.

If Gentran Integration Suite stops, it can be restarted from the persisted WFCs by finding the most recent WFCs and sending those requests to the appropriate services. Internal services can be restarted automatically but external services require user intervention to restart them.

Basic Service Framework

The basic service framework or harness model enables Gentran Integration Suite to view all services similarly. For example, both the Translation service and the File System adapter have harnesses. Although these are different services, they support the same API as represented by the harnesses.

Sometimes an adapter harnesses a system that is outside the control of Gentran Integration Suite (for example, SAP®). Having a harness that presents a consistent interface to the rest of the system is very important. The harness is generic but the adapter is specific to the system with which it interacts to send and receive requests. Using the basic framework, you can start, configure, and stop an adapter for an external system in the Gentran Integration Suite interface. The actual operations of the external system are separate.

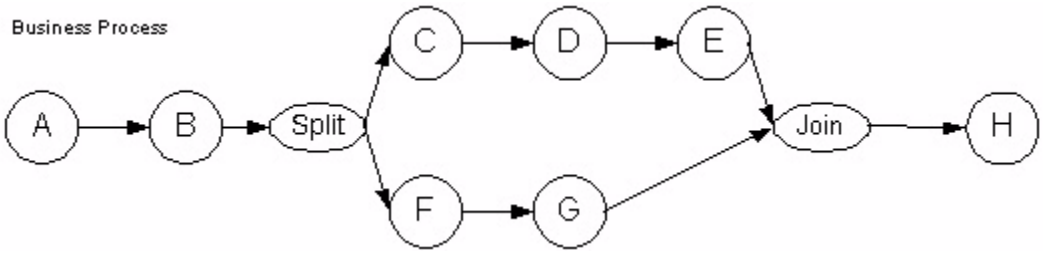
The harness also provides better performance. For example, the harness wrapped around the Translation service caches and reuses translation maps. The actual Translation service is unaffected by this action. This independence is especially important when the wrapped service is outside the control of Gentran Integration Suite.

Split and Join

Through the use of a split, a single business process can have multiple services running simultaneously.

A WFC reflects the status of only one thread or instance of the business process. A join enables the multiple instances to be collected to create a single WFC.

The following figure shows a business process with a split and a join:



Special Service Capabilities

Gentran Integration Suite supports the following unique capabilities, which afford you great flexibility with managing services:

- Large file support – The ability for services to handle files larger than available memory. This can be an effective way to help manage load sharing.
- Service groups – The ability to group “like” services together and treat them as a pool of services
- Storage types – The ability to select the document storage type for a service, such as Database, or File System

For more information about these topics, see the *Service and Adapter Guide*.

Relationship Between Business Processes and Services

Before you can run a business process definition, you must validate and compile it. Validation ensures that all activities in the business process definition are configured appropriately.

Compilation breaks the definition into smaller chunks: a header and entries. The header specifies global properties of the business process definition. Each service within the definition has an entry. The compiled information for each service includes all outbound edges, from that service to subsequent services and the parameters that they require. Outbound edges use service status to direct the flow of execution from one service to another.

Example of Outbound Edges

If Service A returns success or failure, its *success* outbound edge directs processing to continue to Service B, whereas its *failure* outbound edge directs processing to continue to Service X.

Gentran Integration Suite stores the compiled information for each service in the compiled ActivityInfo. ActivityInfo contains an abstract description of the service, such as *Translate the current document using map 5*.

Compilation enables Gentran Integration Suite to predetermine the start node of a business process. This capability makes the business process easy to instantiate and run and prevents Gentran Integration Suite from repeatedly parsing XML. Compilation also reduces the number of database queries because the next-step pointer is stored in the current activity information.

Starting a Business Process

Gentran Integration Suite supports dynamic selection (*bootstrapping*) of business processes. To specify dynamic selection of a business process, configure an adapter to select a business process definition by matching one or more adapter properties. For example, when a user creates a B2B HTTP Server adapter configuration, the user specifies a uniform resource identifier (URI) and then selects either a business process definition or contract specific to that URI. A user can create many B2B HTTP Server adapter configurations with different URIs, each of which invokes a business process or a contract.

Input data enters Gentran Integration Suite through an input adapter. An input adapter performs the following functions:

- Receives data from an external system
- Puts the data and any metadata into an initial workflow context (IWFC)
- Calls the IWFC start method to start the business process, which causes a business process definition to be found and instantiated for the input data

If the input data requires Gentran Integration Suite to start a new process, then the following steps take place:

1. A new WFC is created.
2. The WFC is put in persistent storage.
3. Gentran Integration Suite starts the associated business process.

Many-to-Many Relationship

Gentran Integration Suite bootstrapping creates a many-to-many relationship between adapters and business process definitions. Using the metadata given to the IWFC, one adapter can start several business processes.

Conversely, several adapters can start the same business process. A many-to-many relationship between adapters and business process definitions enables Gentran Integration Suite to focus on business problems, not just on how data arrives.

Making an input adapter the first step in a business process impairs the many-to-many relationship and keeps the business process from being reused as a subprocess.

Business Process Definition Fails to Start

If an adapter tries to start a business process definition that does not exist or is disabled, Gentran Integration Suite saves the request to start the business process definition and any related documents within Gentran Integration Suite. The user can use the business process monitor to view error messages for any business process definitions that failed to execute.

- If the business process definition cannot be found, the user can do an advanced restart and select a different business process definition, which uses the same input data.
- If the business process definition is disabled, then when the user enables that business process definition, Gentran Integration Suite automatically resumes any instances of that business process definition that stopped.

To ensure that an adapter catches the InitialWorkFlowContextException, code its logic accordingly:

```
{
    iwfc.start()
}
  catch (InitialWorkFlowContextException)
 {
 //do not delete our data here if this happens
 //set the appropriate response to the user
 }
```

To enable an adapter to start a business process with more than one document, code the following commands:

```
//for a single document
Document doc = new Document();
etc.
iwfc.putDocument(doc)
//for more than one document
Document doc1 = new Document();
etc.
Document doc2 = new Document();
etc.
iwfc.putDocument(name1, doc2);
iwfc.putDocument(name2, doc2);
```

name1 and *name2* are unique keys for the document within Gentran Integration Suite. When there is only one document, Gentran Integration Suite assigns the unique key of PrimaryDocument.

If a service needs to write more than one document, the service calls:

```
wfc.putDocument(name, doc);
```

For a single document, the service calls:

```
wfc.putDocument(doc))
```

To get a specific document from a set, the service calls:

```
wfc.getDocument(name)
```

Running a Business Process

When a business process starts, the workflow engine (WFE) executes the services defined in the business process definition and the WFE creates a workflow context (WFC) from the initial workflow context (IWFC). The WFE uses the compiled ActivityInfo to get information about the first service to call. Next, the WFE puts the WFC on the JMS queue, so that the client initiating the business process does not wait for it to complete.

The WFE analyzes the compiled information to determine the current activity (service) that needs to be run. This information is stored in the compiled ActivityInfo. The ActivityInfo contains an abstract description of the service.

The WFE determines, for example, how the Translation service has been configured to run.

The following figure shows the execution cycle:

The JMS queue acts as a hand-off point. It does more than routing—it guarantees that the Java thread of execution is not interrupted during the running of the business process. The listener attached to the JMS queue is a lightweight activity engine. The activity engine takes the WFC off the JMS queue and invokes the service. Logically part of the workflow engine, the activity engine calls the service, takes the results from the service, and immediately starts the next cycle, determining the service that needs to be called and requesting that service on the JMS queue.

When configured properly, the usejms property of Gentran Integration Suite enables the activity engine to cycle-call the service directly. Cycle calling avoids the cost of using the JMS queue. However, cycle-calling also limits the clustering capabilities of application servers.

The activity engine can determine the next service because the harness has analyzed the service results and set the state values in the WFC. The activity engine consults these values to determine the next activity. The activity engine uses the return code from the current service to choose the next activity from a set of potential activities listed in the current ActivityInfo.

GIS Components and a J2EE Environment

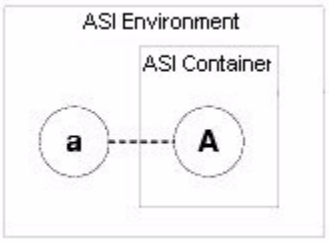
Gentran Integration Suite is written in Java and runs in a container proprietary to Sterling Commerce. This container allows Gentran Integration Suite to be independent of, yet integrate with popular J2EE application servers through standard means such as EJB and JCA clients. For the development of adapters, the service architecture provides a clear separation of concerns. There are two parts to each Adapter Implementation; the Service Harness Implementation that provides interaction and interface with the workflow engine, and the Service Adapter Implementation, which provides the interface to the external system. This allows the Service Adapter Implementation to run independently of the container if necessary, and gives more implementation options for the developer. For services that have no need to interface beyond the process boundary of Gentran Integration Suite, the implementation can be done with the harness.

Service Harness Implementation, Service Adapter Implementation

To overcome these restrictions, Gentran Integration Suite adapters are composed of two parts: *Service Harness Implementation*, the part of the adapter inside the ASI container; and *Service Adapter Implementation*, the part outside the ASI container. Services do not have a Service Adapter Implementation component.

The following figure shows an adapter implemented as Service Harness Implementation and Service Adapter Implementation:

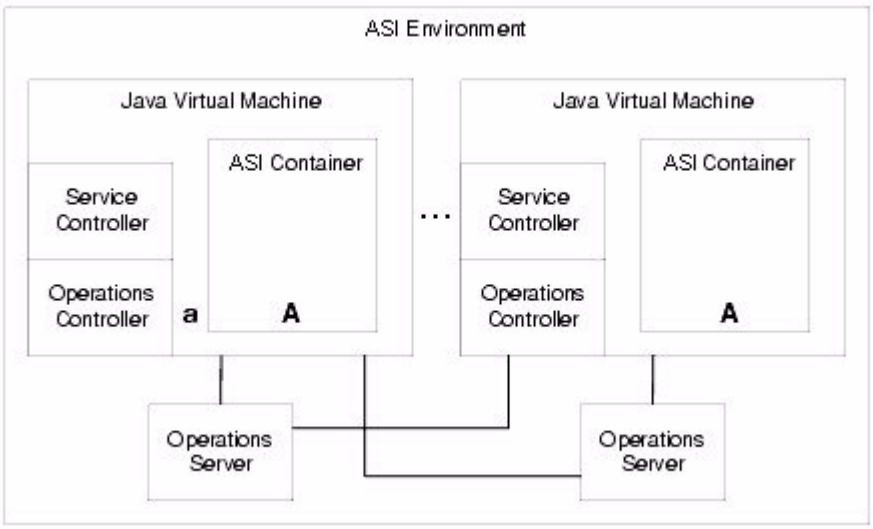
The Service Harness Implementation automatically scales and is portable across clusters because it is instantiated from within the ASI container. The Service Adapter Implementation is tied to a specific computer. The Service Harness Implementation can even move around from one call to the next. The Service Adapter Implementation, however, is fixed next to the resource it is accessing.



Example of Service Harness Implementation and Service Adapter Implementation

A cluster of computers in a ASI environment has private disk space on one computer. The Service Adapter Implementation portion of the File System adapter must be on the computer that can access the disk. The Service Harness Implementation portion of the File System adapter, however, can run in any container on any computer.

In the following figure, the Service Harness Implementation is moved to a different container in a different Java VM:



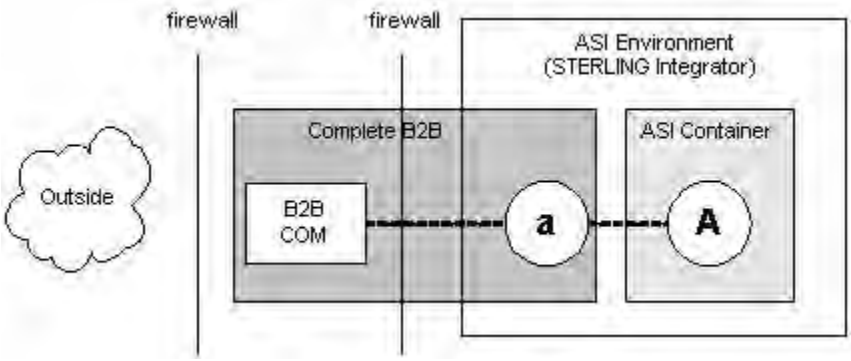
B2B Server

Gentran Integration Suite includes a business-to-business (B2B) server. The B2B server can be viewed as an independent system.

The following figure shows a traditional model of B2B and enterprise application integration (EAI):

However, within Gentran Integration Suite it is more appropriate to view the B2B server as a complex adapter. The B2B server has a two-part Service Adapter Implementation. One part runs in the demilitarized zone (DMZ) and one part runs in the ASI environment.

The following figure shows a B2B server as a complex adapter:



Demilitarized Zone (DMZ)

The part of the B2B server that runs in the DMZ performs communications activities only. It stores no data. Trading profiles are stored in the secure area where Gentran Integration Suite resides. The part of the B2B server in the DMZ can run in a simple Java Virtual Machine (JVM) or a complete ASI environment.

The part of the B2B server in the DMZ and the part of the B2B server inside the secure area communicate as if they were separate systems and not part of a single ASI environment.

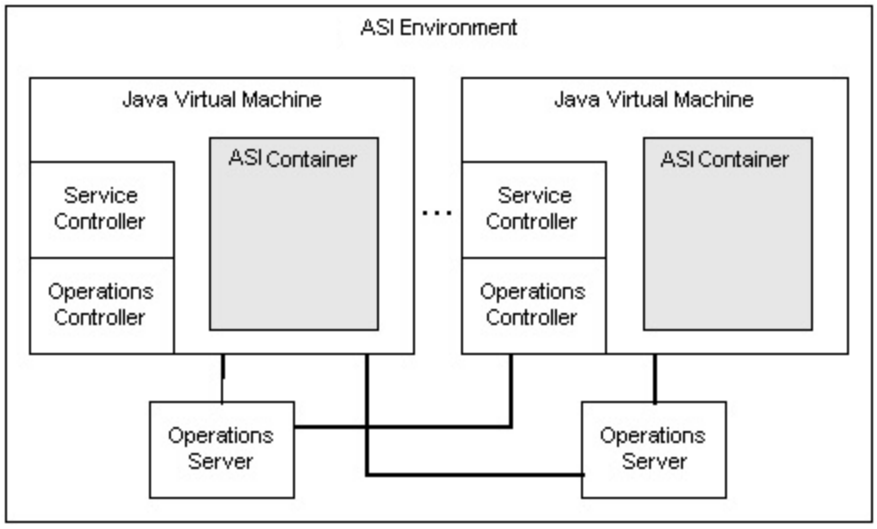
Service and Operations Controllers

The Gentran Integration Suite service and operations controllers monitor and manage executing services and workflows within the Gentran Integration Suite environment. These controllers free system operators and business analysts from having to attend to application-server details.

Service controllers provide a single place within a VM to manage, configure, query, and cache all service-related information. They also enable Gentran Integration Suite to scale and manage the Service Adapter Implementation parts of adapters. There is one service controller per VM in the ASI Container.

Operations controllers manage resources across VM boundaries. You can have multiple operations servers for redundancy and several embedded components, one per VM. Operations servers provide a single point of contact for all operational questions.

The following figure shows the service and operations controllers:



Chapter 2

Using the Service SDK

To help you create services and adapters, Gentran Integration Suite offers the Service Software Developer's Kit (SDK). The Service SDK is a stand-alone graphical interface tool.

Gentran Integration Suite can perform most common tasks needed by a user, but there are times when some piece of functionality is needed that is not provided by the system, and a pre-built service is not available. This is most commonly found in environments that have many legacy systems that do not use standards-based methods for communication. There are several ways to interact with such systems, including the Command Line adapter and the Scripting adapter, but the capabilities of these adapters are limited.

The Service SDK, however, provides the tightest integration with Gentran Integration Suite and enables you to create very complex and complete services and adapters. These services use the same APIs as services included with Gentran Integration Suite, so all of the benefits and infrastructure of Gentran Integration Suite are available. This flexibility and tight integration means that the task of creating an adapter is far more advanced and requires a good knowledge of Java development, the Gentran Integration Suite APIs, and the APIs of any system that will be accessed. The debugging and development of a custom service is more complex than the other two methods, but the benefits are substantial. This method should be used for mission-critical tasks or implementations that are too complex or resource-intensive for other methods.

Note: Because all adapters are a type of service, this guide uses the term *service* for both services and adapters. This guide uses the term *adapter* when the information is unique to adapters.

The chapter covers the following topics:

- Installing the Service SDK
- Starting the Service SDK
- Uninstalling the Service SDK
- Creating and Installing a Service

Note: When developing a custom service, you can manually build one without using the SDK wizards. To do this, you have to create the files in the table in *Files Created by Service SDK* on page 20 and place them in the appropriate directories.

- Compiling a Service SDK Project Outside of the Service SDK

Installing the Service SDK

The *Service SDK* is a Web-deployed application included with Gentran Integration Suite. The SDK is not automatically installed with Gentran Integration Suite; you must install the SDK on your client computer after you have installed Gentran Integration Suite. After the resource files are available on your client computer, you can start using the SDK.

Java Web Start is the tool Gentran Integration Suite uses to deploy the SDK. When you open the SDK, Java Web Start checks the Web server to see whether a new version of the SDK is available. When a new version is available, Java Web Start automatically updates the files on the client computer and opens the SDK.

Requirements for Using the Service SDK

Before you develop a service using the Service SDK:

1. You should have thorough knowledge of and experience with the Java programming language.
2. Install the application server, according to its documentation. Currently, Gentran Integration Suite supports BEA WebLogic® and JBoss™.
3. If you are creating an adapter, set up the system with which the adapter will interact.
4. On the client computer where you are installing the Service SDK:
   - Install the Java Development Kit (JDK) 1.3 or later.
   - Install Java 2 Platform, Enterprise Edition (J2EE) 1.2.
   - Have access to Gentran Integration Suite.

Installation of the Service SDK

To install the Service SDK:

1. From the **Deployment** menu, select **Adapter Utilities** > **Service SDK**.
2. If you have not already downloaded Java Web Start, do so now. In the Service SDK Tool section next to Java Web Start, click **Download**.
3. In the **File Download** dialog box, select the **Save this program to disk** option and click **OK**.
4. In the **Save as** dialog box, select a directory on your client computer and click **Save** to begin downloading Java Web Start.

Note: The download may require several minutes, depending on the speed of your connection.

5. When the download completes, close the **Download** dialog box if it remains open.
6. Open the directory where you downloaded the file for Java Web Start.

7. Double-click the file **javaws-1_0_1_02-win-int-rt.exe** to begin the installation process.
8. After reading the license agreement, click **Accept**.
9. On the **Installation Directory** dialog box, either accept the default directory or click **Browse** to select some other directory to install Java Web Start, and then click **Next**.
10. When you receive a message that setup was unable to detect a usable Java 2 Runtime Environment, either accept the default installation directory or click **Browse** to select some other installation directory, and then click **OK**.

    You can now start the Service SDK.

Starting the Service SDK

You can start the Service SDK from within Gentran Integration Suite or from your computer. When you do start the SDK, a **Service SDK Launch Status** screen displays information messages relating to the SDK starting.

Note: Initial startup may require several minutes, depending on the speed of your connection.

Starting the SDK through Gentran Integration Suite

To start the SDK online through Gentran Integration Suite (and Java Web Start):

1. From the **Deployment** menu, select **Adapter Utilities** > **Service SDK**.
2. In the Service SDK Tool section, click **Go!**

Starting the SDK through Windows

To start the SDK offline through Java Web Start:

1. From the Windows **Start** menu, select **Programs** > **Java Web Start** > **Java Web Start**.
2. From the **View** menu, select **Downloaded Applications** to view installed applications (or installed copies of the SDK).
3. Select the **AdapterSDK** application and click **Start**.

Uninstalling the Service SDK

If you have several copies of the Service SDK installed—for example, you have installed a copy to use for testing and a copy to use for production—you can remove a copy of the SDK using the Java Web Start Manager.

When you need to remove the SDK and the resource files associated with the SDK, you must remove all copies of the SDK from the Java Web Start Manager, and then remove both Java 2 Runtime Environment and Java Web Start.

Removing a Copy of the SDK

To remove a copy of the SDK from the Java Web Start Manager:

1. On the computer where the SDK is installed, from the Windows **Start** menu, select **Programs** > **Java Web Start** > **Java Web Start**.
2. From the **View** menu, select **Downloaded Applications** to view the SDK application (or installed copies of the SDK).
3. Select the SDK (or the copy of the SDK) that you want to remove from the application window.
4. From the **Application** menu, select **Remove Applications**.

Removing the SDK and Resource Files

To remove the SDK and the resources files associated with the SDK:

1. On the computer where the SDK is installed, from the Windows **Start** menu, select **Settings** > **Control Panel**.
2. Double-click **Add/Remove Programs**.
3. In the **Add/Remove Programs Properties** dialog box, select **Java 2 Runtime Environment Standard Edition v1.3.0_02**, and click **Add/Remove**.
4. Follow the remaining instructions to complete the uninstall process.

Note: See the Microsoft® Windows® documentation for complete instructions.

5. Return to the **Add/Remove Programs Properties** dialog box, select **Java Web Start**, and repeat steps 3 through 4.

Creating and Installing a Service

Complete the following tasks to create and install a service for Gentran Integration Suite using the Service SDK:

1. In the Service SDK, open a new project.
2. Create template source code using the Project wizard.

   For more information about creating templates, see *Creating a Template Source Code* on page 19.
3. Add methods to the adapter (this does not apply to services).

   For more information about methods, see *Managing Methods* on page 21.
4. Add parameter groups to the service.

   For more information about parameters groups, see *Managing Parameter Groups* on page 22.
5. Add parameters to the parameter groups.

   For more information about adding parameters, see *Managing Service Parameters* on page 23.
6. Add resources to the service.

   For more information about resources, see *Managing Resources* on page 24.
7. Finish customizing the template source code.

   For more information about customizing templates, see *Customizing the Service Source Code* on page 26.
8. Package the service.

   For more information about packages, see *Generating a Service Package* on page 27.
9. Install the service to a Gentran Integration Suite test environment and verify that the service works.

   For more information about installing services, see *Installing a Service into Gentran Integration Suite* on page 28.
10. Install the service to the Gentran Integration Suite production environment.

Creating a Template Source Code

The Service SDK generates a template source code for your service. The files that the Service SDK generates for you help you get started creating a service.

To create a template source code:

1. In the Gentran Integration Suite Service SDK, click **File** > **New Project**.
2. In the **Project name** field, type a name for the project.

3. Browse to the location where you want to save the project or type the path in the **Project path** field and click **Next**.

> **Note:** If the computer that you are browsing on has many drives mapped, browsing may be slow.

4. In the **Service name** field, type a unique name for the service, according to Java class naming standards. Gentran Integration Suite uses this name internally to reference services.
5. In the **Service package** field, type the name of the Java package where the service should be stored.
6. In the **Service label** field, type the name of the service as it should appear in the Gentran Integration Suite interface.
7. In the **Service description** field, type a description for the service as it should appear in the Gentran Integration Suite interface.
8. If this service is an adapter, select the **Adapter** check box.
9. If you want the adapter to have one object for each configuration of the adapter, select the **Maintain state** check box. If you want the adapter to have one object for all configurations of the adapter, leave the **Maintain state** check box cleared.
10. Click **Finish** to generate the source code. The Service SDK saves the service project with a .sdk ending in the directory you specified and opens the service project.

Service Directory Structure

The Service SDK creates the following directory structure in the location you specify when you create a new service:

- Files
- Servicedefs

Files Created by Service SDK

The following table describes the contents of the files the Service SDK creates in the location you specified:

File	Contents
serviceImpl.java	Service Harness Implementation portion of the adapter. Contains the required processData() function for interaction with the activity engine.
servicenameServer.java	RMI remote interface portion of the adapter that describes the functions that can be called. This file is created for adapters only.
servicenameServerImpl.java	RMI implementation portion of the adapter. This file is created for adapters only.

File	Contents
servicename.xml	How the user interface is presented by the configuration wizard at run time.
servicename_*LangAbrev*.properties	Language properties in the default language of the computer on which you are working. For example, servicename_en.properties is an English language property file. For more information about reference information, see Chapter 3, *Reference Information for Developing a Service*.

Managing Methods

After you have created an adapter (you cannot add a method to a service), you can add one or more methods.

Creating a Method

To create a method for an adapter:

1. In the navigation pane of the Service SDK, right-click **Service name** and select **Add method**.
2. In the **Service Method** dialog box, type the name of the method. Required.
3. For **Return type**, type the Java type the method returns. Required.
4. For **Method Parameters**, click **Add**.
5. Type the name and Java type of the method parameter. Repeat as necessary.

   Note: You must tab out of the last parameter to include it in the method.

6. If you want to remove a method parameter, highlight it, and click **Remove**.
7. When you are finished, click **OK** to add the method.

   The method prototype is placed in the *adaptername*Server.java file. The method is placed in the *adaptername*ServereImpl.java file.
8. Click **File** > **Save project** to save your work.

Deleting a Method

To delete a method:

1. In the navigation pane of the Service SDK, right-click the method you want to delete and select **Delete**.

> ***Caution:*** If you click **Delete**, you cannot undo the deletion.

2. Click **File** > **Save project** to save your work.

Editing a Method

To edit method properties:

1. In the navigation pane of the Service SDK, right-click the method you want to edit and click **Properties**.
2. Edit the method properties as necessary. When you are finished, click **OK**.
3. Click **File** > **Save project** to save your work.

Managing Parameter Groups

After you have created a service, you can add parameter groups. *Parameter groups* are logical groupings of similar parameters, for example, host name and port. Any parameters that you want to add to a service must be added to a parameter group. It is acceptable to have a parameter group with only one parameter. You can also delete parameter groups from the project.

Gentran Integration Suite uses three types of parameter groups: global definition, workflow definition, and instance definition. These are described in more detail in the following procedure.

Creating a Parameter Group

To create a parameter group for the service:

1. In the navigation pane of the Service SDK, right-click **Service name** and select **Add parameter group**.
2. In the **Service Group Properties** dialog box, type the title for the parameter group as you want it to be read by the end user.
3. Select one of the following parameter groups:

Parameter Group	Description	Location for End User
Global Definition	Widest possible scope, applicable to all services of this type.	Gentran Integration Suite interface: Deployment > Services > Installation/Setup

Parameter Group	Description	Location for End User
Workflow Definition	Specific to a single business process.	Graphical Process Modeler
Instance Definition	Specific to a single copy of a service.	Page in a service configuration wizard accessed through Gentran Integration Suite interface, Deployment > Services > Configuration

4. Type any instructions for the end user about what to do with the group of parameters.
5. Click **OK** to add the parameter group.
6. Click **File** > **Save project** to save your work.

Deleting a Parameter Group

To delete a parameter group from the project:

1. In the navigation pane of the Service SDK, right-click the parameter group you want to delete and select **Delete**.

Caution: If you click **Delete**, you cannot undo the deletion.

2. Click **File** > **Save project** to save your work.

Editing a Parameter Group

To edit a parameter group:

1. In the navigation pane of the Service SDK, right-click the parameter group you want to edit and click **Properties**.
2. Edit the parameter group as necessary. When you are finished, click **OK**.
3. Click **File** > **Save project** to save your work.

Managing Service Parameters

After you create a parameter group, you can add parameters to the service.

To create parameters for a service:

1. In the navigation pane of the Service SDK, right-click the parameter group where you want to add a parameter and select **Add service parameter**.

2. Complete the following fields and click **OK** to add a service parameter:

Field	Description
Parameter name	Name of the parameter as it will appear to the user. Required.
Validator	Type of validator, either alphanumeric or number. Optional.
Default value	Default value for the parameter, if there is one. Optional.
Required	If this check box is selected, the parameter is required. Optional.
Java type	Java type of the parameter. Valid options are available in the drop-down list. Required.
HTML type	HTML type of the parameter. Required. ◆ Text – text field ◆ Select – drop-down list ◆ Radio – radio button ◆ Checkbox – check box If you select either **Select** or **Radio**, the **Selection** tab becomes available for you to specify the selection options for the parameter. 1 Click the **Selection** tab. 2 Click **Add**. 3 Type the name and value of the selection. Repeat as necessary. **Note:** You must tab out of the last selection to include it in the parameter. 4 To remove a selection, highlight it, and click **Remove**. 5 When you are finished, click the **Details** tab to complete the parameter.
Size	Number of characters for the parameter display size. Optional.
Max size	Maximum number of characters allowed for the parameter. Optional.

6. Click **File** > **Save project** to save your work.
7. After you make a change in the navigation pane (left side of project), you should regenerate the project. Click **Project** > **Regenerate code**.

Managing Resources

Depending on the service you are creating, you may need to add third-party files such as BPML, maps, scripts, databases, properties, libraries, and .jar files. In addition to adding these resources, you can also remove resources from the project file (the original files are not deleted from their original location). Additionally, you can create folders for any files you want to add to the service project.

The service SDK adds these files in the location you specify when you create the service. If you add any of these resources, you must add the following folders with these exact names, according to the resources you are adding to the service:

- bpml/*servicename*
- User Accounts
  - maps
  - scripts

Note: After you make a change in the navigation pane, you should regenerate the project. Click **Project** > **Regenerate code**.

Creating a Folder

To create a folder in the service project:

1. In the navigation pane of the Service SDK, right-click **resources** and select **Add folder**.
2. In the **Create new directory** dialog box, type the name of the folder.

   The folder appears in the navigation pane under **resources**.
3. Click **File** > **Save project** to save your work.

Adding a File

To add a file to the service project:

1. In the navigation pane of the Service SDK, right-click **resources** and select **Add file**.
2. In the **Add File to Project** dialog box, browse to the location of the file you to want add.
3. Click **Add file**.
4. Click **Project** > **Regenerate code**.
5. Click **File** > **Save project** to save your work.

Removing a Folder or a File

To remove a folder or a file from the service project:

1. In the navigation pane of the Service SDK under **resources**, right-click the file or folder you want to remove and select **Remove**.
2. Verify that you want to remove the file or folder by clicking **Yes**.

   The folder or file is removed from the project.
3. Click **File** > **Save project** to save your work.

Installing a .jar File

If you create your own service or adapter (whether or not you use the Service SDK wizards), you must install the appropriate .jar file.

To install a .jar file into Gentran Integration Suite from the domain directory, enter the following command:

./InstallService.sh *basedir*/*myservice*.jar

Adding Third-Party Resources

To add third-party resources (such as libraries or properties files):

From the directory where Gentran Integration Suite is installed, set the CLASSPATH environment variable to point the library package (that is, the jolt.jar file).

./Install3rdParty.sh *mylibrary -j directory*/jolt.jar

Customizing the Service Source Code

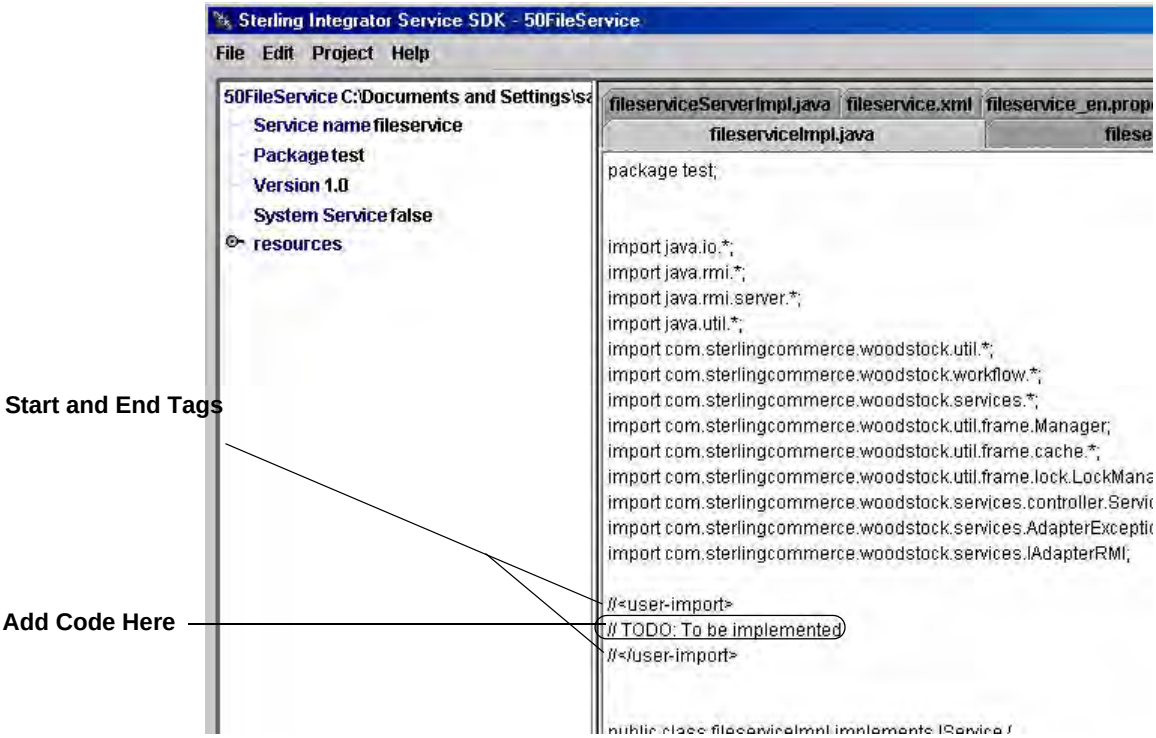
In addition to adding parameter groups and parameters and any other additional files to the service, you also need to further customize the template source code that the Service SDK generates for you.

As you make changes to the template source code files, you do not need to regenerate the code.

To customize the template service source code:

1. Use an IDE or type directly in the content pane to add new material to the files but only wherever TODO place markers are located. In the Service SDK, use the tabs to view the different files. The following figure shows where to add code.

> ***Caution:*** You must keep your material between the start and end tag; otherwise, you could lose material each time you regenerate the code.



2. As you work and when you are finished, save your changes in the source code by clicking **File** > **Save source**.

Generating a Service Package

After you have completed the service, you must package it so that you can install it into Gentran Integration Suite.

To generate a service package:

1. In the Service SDK, click **Project** > **Generate package** or right-click in the navigation pane.
2. Select **Package service**.

The Service SDK packages the service into a file: *servicename*_version.jar, where *servicename* is the name of your service. The packaged service .jar file is located in the root directory where your project is located.

The service is now ready to be installed into Gentran Integration Suite.

Installing a Service into Gentran Integration Suite

After you generate the initial source code with the Service SDK and after you customize and package the source code, you must install the packaged service into Gentran Integration Suite.

Note: Before you install your new packaged service into a Gentran Integration Suite production environment, you should install the service in a Gentran Integration Suite test environment.

Installing into a UNIX Environment

To install a service into a Gentran Integration Suite UNIX environment:

Note: All of the commands in the following procedure are executed from the install/bin directory.

1. Shut down the test Gentran Integration Suite copy, if it is running, by entering the appropriate command:

Scenario	Command to Use
Test environment where it does not matter if you stop running business processes	./hardstop.sh
Production environment or test environment where it does matter if you stop running business processes	./softstop.sh

2. Copy the packaged service to the computer where the test Gentran Integration Suite is installed.
3. If the test Gentran Integration Suite copy uses MySQL™, verify that MySQL is running when you install the service.
4. Enter the following command to run MySQL without starting Gentran Integration Suite:

   ./control_mysql.sh start

5. To install the service, enter the following command:

   ./InstallService.sh *svcpath*

   svcpath is the location of the packaged service that you copied to the computer.

6. To install any third-party libraries, properties, or .jar files, enter the following command:

   ./install3rdParty.sh *vendorName vendorVersion* <-j *jars* | -l *sharedLibs* | -p *property files*> [-nodeploy]

   Here are some examples of the install3rdparty.sh command:

```
./install3rdParty.sh ibm_sap 3_5 -j /usr/local/lib/sap/*.jar
./install3rdParty.sh ibm_sap 3_5 -j /usr/local/lib/sap/infobus.jar
./install3rdParty.sh oracle  1_2 -d /usr/local/lib/jar/*.jar
./install3rdParty.sh oracle  1_2 -d /usr/local/lib/jar/classes12.zip
./install3rdParty.sh ibm_sap 3_5 -j /usr/local/lib/sap/infobus.jar
./install3rdParty.sh ibm_sap 3_5 -l /usr/local/lib/sap/*.so
./install3rdParty.sh ibm_sap 3_5 -l /usr/local/lib/sap/libivjsid35.so
./install3rdParty.sh ibm_sap 3_5 -p /usr/local/lib/sap/*.properties
./install3rdParty.sh ibm_sap 3_5 -p /usr/local/lib/sap/some.properties
./install3rdParty.sh mqseris 2_0 -r /usr/local/lib/mqs/service.properties
```

7. If you started MySQL, stop MySQL by entering the following command:

   ./control_mysql.sh stop

8. Restart Gentran Integration Suite by entering the following command:

   ./run.sh

   After you verify that the service works correctly, install the service into the Gentran Integration Suite production environment using the preceding instructions.

After you have installed a service into Gentran Integration Suite, you can see the icon for the new service in the Graphical Process Modeler (GPM) under the Custom Service stencil. The service is also available through the Gentran Integration Suite interface so users can create a service configuration. In the Gentran Integration Suite interface, the new service is listed with all of the other services and adapters.

Installing into a Windows Environment

To install a service into a Gentran Integration Suite Windows environment:

Note: All of the commands in the following procedure are executed from the install/bin directory.

1. Shut down the test Gentran Integration Suite copy, if it is running, by entering the appropriate command:

Scenario	Command to Use
Test environment where it does not matter if you stop running business processes.	hardstop.cmd
Production environment or test environment where it does matter if you stop running business processes.	softstop.cmd

2. Copy the packaged service to the computer where the test Gentran Integration Suite is installed.
3. If the test Gentran Integration Suite copy uses MySQL, then MySQL must be running when you install the service.
4. Enter the following command to run MySQL without starting Gentran Integration Suite:

   control_mysql.cmd start

5. To install the service, enter the following command:

   InstallService.cmd *svcpath*

   svcpath is the location of the packaged service that you copied to the computer.
6. To install any third-party libraries, properties, or .jar files, enter the following command:

   install3rdParty.cmd *vendorName vendorVersion* <-j *jars* | -l *sharedLibs* | -p *property files*> [-nodeploy]

   Here are some examples of the install3rdparty.cmd command:

```
install3rdParty.cmd ibm_sap 3_5 -j /usr/local/lib/sap/*.jar
install3rdParty.cmd ibm_sap 3_5 -j /usr/local/lib/sap/infobus.jar
install3rdParty.cmd oracle  1_2 -d /usr/local/lib/jar/*.jar
install3rdParty.cmd oracle  1_2 -d /usr/local/lib/jar/classes12.zip
install3rdParty.cmd ibm_sap 3_5 -j /usr/local/lib/sap/infobus.jar
install3rdParty.cmd ibm_sap 3_5 -l /usr/local/lib/sap/*.so
install3rdParty.cmd ibm_sap 3_5 -l /usr/local/lib/sap/libivjsid35.so
install3rdParty.cmd ibm_sap 3_5 -p /usr/local/lib/sap/*.properties
install3rdParty.cmd ibm_sap 3_5 -p /usr/local/lib/sap/some.properties
install3rdParty.cmd mqseris 2_0 -r /usr/local/lib/mqs/service.properties
```

7. If you started MySQL, stop it by entering the following command:

   control_mysql.cmd stop

8. Restart Gentran Integration Suite by entering the following command:

   run.cmd

   After you verify that the service works correctly, install the service into the Gentran Integration Suite production environment using the preceding instructions.

After you have installed a service into Gentran Integration Suite, you can see the icon for the new service in the Graphical Process Modeler (GPM) under the Custom Service stencil. The service is also available through the Gentran Integration Suite interface so users can create a service configuration. In the Gentran Integration Suite interface, the new service is listed with all of the other services and adapters.

Compiling a Service SDK Project Outside of the Service SDK

You can also compile a Service SDK build (to create a service or adapter) using the command line, independent of the SDK user interface. This method is very advantageous because it provides you with copious useful information about the compile and packaging steps of the Service SDK.

The Service SDK uses the Apache Ant build product internally to compile and package builds, and thus you must have Apache Ant installed and configured on your development system to compile a build outside of the Service SDK. For more information about Apache Ant, see http://ant.apache.org/.

The Service SDK wizard generates wrapper code that contains:

- The **build.xml** file, which is an Apache Ant project file.
- The **build.properties** file, which contains important paths, including the classpath used to compile the build.

> **Note:** If external libraries are required to compile custom code, you can add the .jar files to the classpath in the properties file.

To compile a service or adapter outside of the Service SDK user interface:

1. Use the Service SDK user interface to generate the wrapper code. For more information about using the Service SDK, see *Creating and Installing a Service* on page 19.
2. Save your project. For more information about saving, see *Creating a Template Source Code* on page 19.
3. Select **Project > Generate Package** to create the files that are necessary to build the project. For more information about generating a service package, see *Generating a Service Package* on page 27.
4. Using Windows Explorer, navigate to your project directory.
5. Rename the build.xml file as **external_build.xml**.
6. Rename the build.properties files as **external_build.properties**.
7. Open the **external_build.xml** file for editing.

8. Directly below the line containing the project tag, add a new line containing: **<property file=”external_build.properties”/>**

   This is a sample of a modified external_build.xml file:

```
<project name="ServiceSDK" default="dist" basedir=".">
<property file="external_build.properties"/>
<target name="init">
<!-- Create the time stamp -->
<tstamp/>
<delete dir="${srcdir}/dist/${adapter}"/>
<mkdir dir="${srcdir}/dist/${adapter}"/>
<mkdir dir="${srcdir}/dist/${adapter}/ejbs/"/>
<mkdir dir="${srcdir}/dist/${adapter}/jars/si/${version}"/>
<mkdir dir="${srcdir}/classes"/>
</target>
<target name="compile" depends="init">
<!-- Compile the java code from ${src} into ${build} -->
<javac classpath="${classpath}" srcdir="${srcdir}"
destdir="${srcdir}/classes"/>
<!--<javac srcdir="${src}" destdir="${src}/classes"/>-->
</target>
<target name="rmic" depends="compile">
<rmic classpath="${classpath}" base="${srcdir}/classes"
includes="**/*ServerImpl.class"/>
</target>
<target name="dist" depends="rmic">
<copy todir="${srcdir}/dist/${adapter}">
<fileset dir="${srcdir}/resources">
<include name="**/*"/>
</fileset>
</copy>
<jar
destfile="${srcdir}/dist/${adapter}/jars/si/${version}/${adapter}.jar"
basedir="${srcdir}/classes"/>
<jar destfile="${srcdir}/${adapter}_${version}.jar"
basedir="${srcdir}/dist"/>
</target>
</project>
```

9. Save the external_build.xml file and exit the text editor.
10. Using the command line, navigate to the project directory.

11. Execute the build.

    If you have a full version of Apache Ant installed and configured on your development system, you can execute the build by typing the following command:

    ```
    ant -buildfile external_buld.xml
    ```

    If you do *not* have a full version of Apache Ant installed and configured on your development system, you can execute the build by typing the following command:

    ```
    java -classpath
    SterlingCommerce\ServiceSDK\lib\ant.jar -Dant.
    home=\SterlingCommerce\ServiceSDK org.apache.tools.ant.Main
    -buildfile external_build.xml
    ```

    When you execute the build, Apache Ant executes and compiles the necessary files and builds the service package. Any errors that occur in this process are displayed in the command line window.

    If any modifications are necessary, you can directly modify the source files or you can use the Service SDK user interface to regenerate the wrappers. For more information on using the Service SDK, see *Creating and Installing a Service* on page 19. You can then repeat the build process as many times as necessary.

Chapter 3

Reference Information for Developing a Service

In addition to knowing how to use the SDK tool, you need to understand the following concepts, to develop a service for Gentran Integration Suite.

This section covers the following topics:

- Service Architecture Summary
- Workflow Context
- Service Controller
- Error and Status Reporting
- Configuring Services
- Logging Service

Service Architecture Summary

An adapter interacts with external systems to get data in and out of Gentran Integration Suite. Typically, an adapter consists of a harness, a Remote Method Invocation (RMI), and files that enable the adapter to be used in the Gentran Integration Suite interface.

Harness

The harness part of the adapter must implement the processData() function, which the activity engine calls whenever it has work for the adapter to perform. This function can be called to push data out of the system or signal that data needs to be collected.

RMI (Service Adapter Implementation)

Preferably, most of an adapter workload should reside in the Service Harness Implementation. The Service Implementation exists mainly to allow for a separation of concerns from the harness.

Typically, adapters do some work that would be inefficient or hindering operating completely within the ASI Container. For example, the service implementation can be set up to wait for data to arrive and to periodically poll the external system for data.

Service XML File and Language Files

For a service to appear in the Gentran Integration Suite user interface and to be configured, XML entries must be added to the service.xml file with its associated language file.

For more information about language properties, see *Language-Specific Properties Files* on page 45 and *Service XML File* on page 45.

Scalability

Services in Gentran Integration Suite are scalable: For adapters, the Service Implementation creates a new thread to service each new request it receives. The Service Implementation is multi-threaded by default. You do not need to write additional code.

Workflow Context

The *workflow context API* encapsulates a basic unit of work, including all parameters required by the adapter to act on that unit of work.

The workflow context maintains the state of the business process from service to service. It contains, among other things, the document being manipulated by the business process. This is also where each service reports errors and status. The Gentran Integration Suite infrastructure is designed to persist the workflow context between steps.

The workflow context contains several components:

- *Input Parameters* – Retrieve parameters before beginning the operation
- *Workflow Document Body* – Set up the document body
- *Error Reporting* – Set up status and error reporting

Input Parameters

A service should have all of its parameters before doing any of its core logic. The workflow context provides the getWFContent method to retrieve parameters:

```
getWFContent(String parmName);
```

The getWFContent method retrieves a named input parameter from the workflow context. It gets global (service type), copy (service configuration), and WFD (workflow definition or business process definition) level parameters.

WFD parameters override service configuration parameters, which override service type parameters. If the workflow content message contains a string value with the parmName requested in the getWFContent method call, then the value in the workflow content hash table overrides all other values.

The hash table can contain any type of object. The getWFContent method enables a parameter to override a higher-level parameter only if its value is stored as a string in the hash table.

For example, the following string would retrieve an input parameter named *URL*:

```
String url = getWFContent("URL");
```

The service may need to get a parameter passed at run time by a previous service. If the parameter is not a service type, service configuration, or WFD parameter being overridden, then the service calls:

```
getWFContent(parmName);
```

Workflow Document Body

A typical service or adapter operates on a document contained within the workflow context. The document contains the body of the document, information about the name of the body, and metadata that describes the document.

To retrieve the document from the workflow context, use the getPrimaryDocument method:

```
getPrimaryDocument(document);
```

Here is an example:

```
Document doc = wfc.getPrimaryDocument();
if(doc == null) {  // no document?
  theAdapterLog.logError("Required document not found");
  wfc.setBasicStatusError();
  sci.unregisterThread();
  return wfc;
} // end if

byte[] body = doc.getBody();
```

After the document is retrieved, you can obtain the body of the document by using the getBody method:

```
getBody(body);
```

The following example shows a new document body inserted into the workflow context:

```
Document document = wfc.createDocument();
document.setBody(body);
document.setBodyName("somename");
wfc.putPrimaryDocument(document);
```

Error Reporting

Another important component of the workflow context is status and error reporting. The workflow engine requires an adapter to return status information at the completion of the requested activity. The requesting business process uses this information to make decisions that control business process flow. Status is reported within the workflow context.

For more information about error reporting, see *Error and Status Reporting* on page 42.

Service Controller

The *service controller* is a unified framework that all adapters use to remove application-server dependencies.

The service controller is also responsible for starting and stopping the adapter.

Stateless and Stateful Adapters

Stateless and stateful adapters differ at the object level. For stateless adapters, the service controller instantiates one object that services all configured copies of the adapter. Each request to the Service Implementation of the adapter must be a complete request, because states cannot be maintained between requests. For stateful adapters, the service controller instantiates one object for each configured copy of the adapter.

Instance variables for RMI objects are not useful because multiple threads (or, in the case of stateless adapters, multiple copies of the adapter) have access to the same instance variables, as if they were class variables.

Method variables are unique to the invocation of the method, so they are acceptable to use.

Service Controller Interface

An adapter is composed of two parts:

- A harness that is the interface to the workflow engine
- An RMI server that communicates with external systems

The following code example shows how the adapter processData method:

1. Registers with the service controller

2. Finds its RMI service
3. Invokes an RMI method
4. Unregisters with the service controller
5. Returns to the following code to the workflow engine:

```
String svcName = wfc.getServiceName;
ServicesControllerImpl sci = ServicesControllerImpl.getInstance();
sci.harnessRegister(new Integer(wfc.getWorkFlowId()).toString(),
  svcName);
        try{
            rmi = (Yourserver)sci.getAdapter(svcName);
        }
        catch(Exception e){
            wfc.setBasicStatusError();
            sci.unregisterThread();
            return wfc;
        }
        if ( rmi == null ){
            wfc.setBasicStatusError();
            sci.unregisterThread();
            return wfc;
        }
        try {
  // request “Service Adapter Implementation” to do work
         rmi.someRMIMethod(parms, xmlInBytes);
        }
        catch(Exception e) {
            wfc.setBasicStatusError();
            sci.unregisterThread();
            return wfc;
        }
```

This code example uses the following methods to accomplish its work:

Method	Description
harnessRegister(workflowID, serviceName)	Assists the service controller in its monitoring and control functions. The harnessRegister should be called at the beginning of the processData method. Returns – None.
UnregisterThread()	Assists the service controller in its monitoring and control functions. It should be called before the return of the processData method. Returns – None.

Service Controller Interface – RMI

The IAdapterImpl class provides the following methods for use in the Service Implementation of the adapter:

Method	Description
startup()	The service controller calls startup() after a stateful adapter is created. Startup() should perform all setup and initialization necessary for the adapter to function correctly. This method returns a Boolean value. A true return indicates that startup was successful.
shutdown()	The service controller calls the shutdown() method when a stateful adapter shutdown is required. shutdown() should perform all operations necessary to shut down the adapter. In the case of a multi-threaded adapter, shutdown() must ensure that all of its threads are stopped before returning to the service controller. shutdown() should wait for threads that are performing work directly for a workflow to become quiescent. The adapter can stop threads that are waiting for external input. The IAdapterImpl base class provides the methods interruptThreads() and stopThreads() to assist in shutting down errant threads. As an additional assistance to the shutdown() method, the base class provides a count of registered threads. This count can be found in the invokes variable. The shutdown() method should poll this variable no more than once a second, waiting for it to decrement to zero. Note that the shutdown thread does not appear in this count. If the adapter has threads that listen on sockets, the invokes count may never go to zero. The shutdown() method will need to account for this case. The service controller will not wait indefinitely for shutdown() to fulfill its responsibilities. The adapter can configure the time-out period by overriding getShutdownTimeout(). The service controller enables shutdown() the amount of time returned from getShutdownTimeout(), and then the shutdown thread is terminated. This method returns a Boolean value. A true return indicates that the shutdown() method completed successfully.

Method	Description
refresh()	The service controller calls refresh() when the configuration changes for a stateful adapter. refresh() should perform all setup and initialization necessary for the adapter to function correctly. If an adapter maintains a connection or connections to an end system, and the connectivity configuration changes, refresh() should return a false value to the service controller. In this case, the service controller shuts down the adapter and then restarts it. This action prevents workflows from trying to invoke the adapter while connectivity changes are occurring. This method returns a Boolean value. A true return indicates that refresh() completed successfully. A false return indicates that the adapter did not refresh, in which case the service controller shuts down the adapter and then restarts it.
getShutdownTimeout()	The service controller calls getShutdownTimeout() to determine how long to wait for shutdown to complete before terminating the adapter threads. A default implementation provided in the base class returns 60 seconds. This method returns the period for shutdown to complete, in milliseconds.
interruptThreads()	The interruptThreads method is provided in the base class to assist the adapter shutdown() method in shutting down its active threads. interruptThreads() calls the interrupt method on each active thread and assists in a graceful shutdown where possible. Returns – None.
stopThreads()	The stopThreads method is provided in the base class to assist the adapter shutdown() method for active threads. stopThreads() terminates every active thread. Returns – None.
registerThread()	The registerThread method assists the service controller in its monitoring and control functions. registerThread() should be called at the beginning of each method called by the Service Harness Implementation portion of the adapter. Returns – None.

Method	Description
unregisterThread()	The unregisterThread method assists the service controller in its monitoring and control functions. unregisterThread() should be called at the end of each method called by the Service Harness Implementation portion of the adapter. Returns – None.

Error and Status Reporting

The workflow engine requires a service to return status information at the completion of the requested activity. The requesting business process uses this information to make decisions that control business process flow.

Three types of status information are returned to the workflow engine:

- Basic status
- Advanced status
- Exceptions

Basic Status

Basic status is the overall status of the work performed by the service.

```
setBasicStatus(status)
```

For example:

```
setBasicStatus(com.sterlingcommerce.woodstock.workflow.WorkFlowContext.ERROR)
;
```

Advanced Status

The *advanced status* is a modifier for the basic status. Reporting the advanced status requires the service writer to analyze the service error categories. Business analysts use the list of advanced errors to test for error conditions. The list should be representative, but not long.

The workflow context provides the following method for setting advanced status:

```
setAdvancedStatus(String advancedStatus);
```

Exceptions

An *exception* indicates a possible failure condition. A service may need to generate a workflow exception when a required input parameter is:

- Missing
- Invalid
- Disabled. For example, a map or a workflow definition is disabled.

Use the following syntax to construct an exception:

```
new WorkFlowException(String errorText, int reasonCode)
```

Workflow exception reason codes are:

- public final static int GENERAL_PARM_ERROR = 0; (Default)
  Use this in situations that require, for example, missing properties files.
- public final static int MANDATORY_PARM_MISSING = 1;
- public final static int INVALID_VALUE_FOR_PARM = 2;
- public final static int RESOURCE_DISABLED = 3;
- public final static int NO_DOCUMENT = 4

The workflow engine places the error text string for the workflow exception into the status report.

Status Report

The service can add text describing the activity performed to the workflow context. The workflow context uses the following method for this purpose:

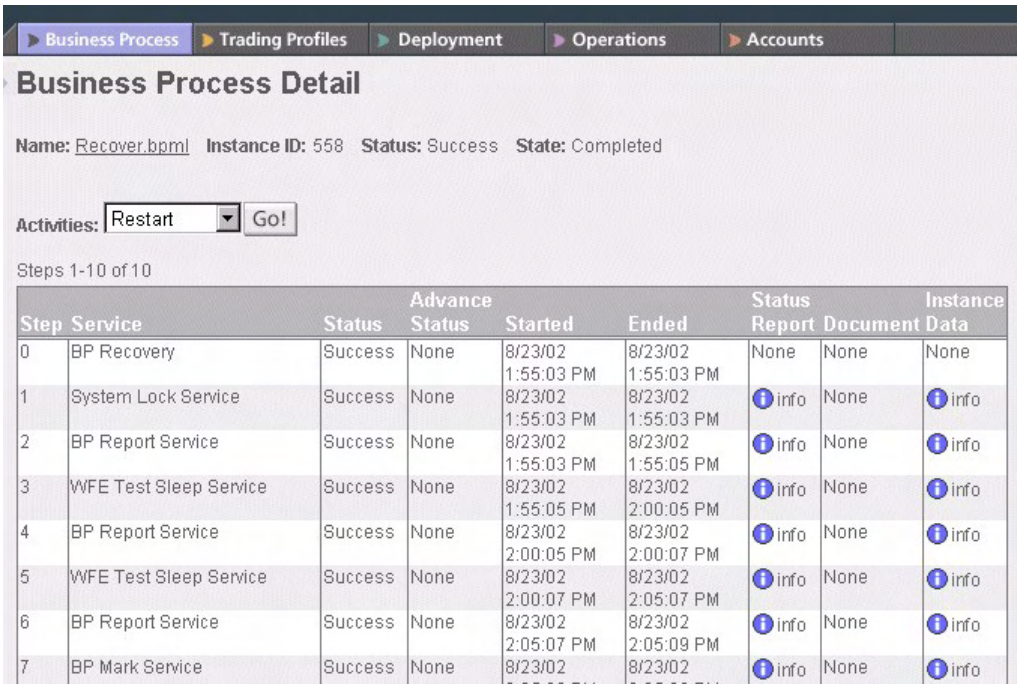
```
setWFStatusRpt(“Status_Report”, String statusReportText);
```

A status report is available to view if an Info icon appears in the Report column. Click the icon to display the status report for that service. You can view the status report text from the Gentran Integration Suite interface.

Successful Invocation

Use the Gentran Integration Suite interface to view the progress of a business process as it executes; use the business process monitor to view details after a business process has run. When a service is successfully invoked, the Status column displays *Success*, as shown in the following figure. This display is a result of calling the setBasicStatusSuccess() method in the workflow context. In this case, the advanced status does not need to be set.

The following figure shows the business process detail page:



Step	Service	Status	Advance Status	Started	Ended	Status Report	Document	Instance Data
0	BP Recovery	Success	None	8/23/02 1:55:03 PM	8/23/02 1:55:03 PM	None	None	None
1	System Lock Service	Success	None	8/23/02 1:55:03 PM	8/23/02 1:55:03 PM	info	None	info
2	BP Report Service	Success	None	8/23/02 1:55:03 PM	8/23/02 1:55:05 PM	info	None	info
3	WFE Test Sleep Service	Success	None	8/23/02 1:55:05 PM	8/23/02 2:00:05 PM	info	None	info
4	BP Report Service	Success	None	8/23/02 2:00:05 PM	8/23/02 2:00:07 PM	info	None	info
5	WFE Test Sleep Service	Success	None	8/23/02 2:00:07 PM	8/23/02 2:05:07 PM	info	None	info
6	BP Report Service	Success	None	8/23/02 2:05:07 PM	8/23/02 2:05:09 PM	info	None	info
7	BP Mark Service	Success	None	8/23/02	8/23/02	info	None	info

Unsuccessful Invocation

An unsuccessful invocation of an adapter or service should result in the Status column displaying *Error*. This display is a result of calling the setBasicStatusFailure() method in the workflow context. The setAdvancedStatus() method is also called to give additional information about the failure condition.

Configuring Services

The Gentran Integration Suite console can be set up to prompt for configuration information specific to the service being developed. Simple service configuration does not require you to write any Java code. The only requirement is that the service XML file is set up to specify the information to be collected.

To set up the Gentran Integration Suite console, you must use the following files:

- Language-specific properties files – Provide screen text in the language chosen by the user.
- Service XML file – Describes the information collected at configuration time.

Language-Specific Properties Files

The language-specific files should exist for each service XML file. It is also the custom to pick a two- or three-character service abbreviation for the service and prepend this abbreviation to each language-specific property name. For example, for a file system, *fs* would indicate file system and look like *fs.label* or *fs.description*. This convention helps to guarantee that the language-specific property names are unique.

Here is an example of a language-specific properties file:

```
fs.label = File System Adapter
fs.description = Collects and Extracts files from a file system.
fs.wfd.group1.title = Workflow Properties
fs.wfd.group1.instructions = Specify the appropriate workflow settings.
fs.instance.group1.title = Collection
fs.instance.group1.instructions = Specify the appropriate settings for
collecting data using the File System Adapter.
fs.action = Action
fs.cfolder = Collection Folder name
fs.efolder = Extraction Folder name
fs.pollinterval = Poll Interval (mins)
```

Service XML File

Here is an example of the service XML file:

```
SERVICE name="FileSystem" description="fs.description" label="fs.label"
implementationType="CLASS" JNDIName="FileSystemEJBHome" type="Adapter"
adapterType="STATELESS"
adapterClass="com.sterlingcommerce.woodstock.services.filesystem.FileSystemSe
rverImpl" version="1.0" SystemService="NO">
```

The following table describes the variables in the service XML file:

Variable	Description
name	Descriptive name of the service.
description	Description in language-specific form.
label	Descriptive name in language-specific form.
implementationType	Either RMI or CLASS.
JNDIName	Lookup name of the service.
type	Either Adapter, basic, Advanced, Split, or Join.
adapterType	Either STATEFUL or STATELESS.
adapterClass	Class name of the Service Implementation of the adapter.
version	1.0
systemService	NO for adapters.

<VARS> Tags

Inside the <Service> tag are <VARS> tags. The <VARS> tags contain definitions of configuration items to collect from the user. Three types of <VARS> tags correspond to the scope of the configuration:

- global – The widest possible scope, applicable to all adapters of this type. Configuration parameters are displayed in the **Deployment > Services > Installation/Setup** section.
- instance – Limited in scope to a single instance of an adapter. Configuration parameters are displayed in the **Deployment > Services > Configuration** section.
- wfd – Workflow definition configuration for use only by the Graphical Process Modeler. The primary purpose of this tag type is to define the possible configuration that can be made in the workflow definition. Because the configuration is defined here, the Graphical Process Modeler can display the possible configuration to the user.

<VARS> Tag Example

Following is a sample <VARS> tag:

```
<VARS type="instance">
```

<GROUP> Tags

Inside the <VARS> tags are <GROUP> tags. <GROUP> tags group configuration information by page. The <GROUP> tag is part of the <VARS> tag and contains title and instructions.

- title – The title of the current page
- instructions – Help about what the user is supposed to do with the current page.

<Group> Tags Example

Following is a sample <Group> tag:

```
<GROUP title="fs.instance.group1.title"
instructions="fs.instance.group1.instructions">
```

<VARDEF> Tag

Inside the <GROUP> tag is the <VARDEF> tag.

The following table describes the <VARDEF> tag elements:

Tag Type	Description
varname	Name of the Java property that a service uses to retrieve data collected from the user.
type	Type of the input, normally String.
htmlType	Type of input you are retrieving from the user. Normal textual information is htmlType *text*, radio buttons are specified with *radio*, and password information can be retrieved with the htmlType *Password*. For text area, specify *textarea*. For drop-down list, specify the htmlType *select*.
validator	Validation types—for example, ALPHANUMERIC specifies that only alphabetic and numeric characters are accepted as input. NUMBER and NUMERIC specify only numeric validation.
label	Description of the configuration to be entered by the user.
size	Field size that displays to the user when collecting information.
maxsize	Maximum number of characters the user can enter for this variable.
required=NO	Specified only if the variable is not required.
defaultVal	Default value that is useful only if the VARS type is wfd.

The following example shows how all of the tags work together:

```
<VARS type="instance">
    <GROUP title="fs.instance.group1.title"
instructions="fs.instance.group1.instructions">
<VARDEF varname="collectionFolder" type="String" htmlType="text"
validator="ALPHANUMERIC" size="30" maxsize="250" label="fs.cfolder" />
     <VARDEF varname="useSubFolder" type="String" htmlType="radio"
validator="ALPHANUMERIC" options="radio2" label="fs.subfolders" />
    </GROUP>
</VARS>
```

Logging Service

This section provides general guidelines for logging from a service in the Gentran Integration Suite framework. The *Logging service* is a unified framework for logging messages.

Because services may have different needs and functions, apply the following guidelines only with a good understanding of the service logging output.

Logging Event Guidelines

The following table lists the guidelines for the types of events to log for each logging method:

Method	Guideline
Error	Log any non-exception occurrence that would cause the intended operation of the service to fail. Examples ◆ Communication failed to external system ◆ Invalid input data
Exception	Log any checked exceptions that are the result of an error or fault. Exceptions that are handled by the service should not be logged with logException. Example Any caught exception that is not handled by the adapter

Method	Guideline
Warn	Log any system disruption that is neither fatal nor a handled exception. Also log any notifications of possible error conditions. Examples ◆ Connection to external resource is lost while no processing is occurring ◆ Disk space limits ◆ License file expirations
Debug	Log anything that is useful information for development/debugging purposes, including general processing information and functionally relevant occurrences. Examples ◆ Entering and exiting of major methods ◆ Method input parameters ◆ Method return values ◆ Parameter values (setting and getting) ◆ Initiation and completion of operations
Log	Log anything that should always appear in the log file. This method sends messages to the default log regardless of log level. Examples ◆ Adapter StartUp ◆ Adapter ShutDown ◆ Adapter Refresh ◆ Connection to external systems

XLogger Logging

The XLogger class provides a unified log output, which includes the following:

- Module name
- Thread ID
- Service name

This class can also be used in stateful RMI adapters (Service Adapter Implementation's).

For more information about RMI logging, see *RMI Logging* on page 50.

EJB Logging

In the Service Harness Implementation of the adapter, the service name is available in the workflow context and can be used in the constructor when creating an XLogger copy.

For example, use this form of the constructor at the beginning of the processData method:

```
adapterLogger = new XLogger(classsName, getServiceName());
```

The adapterLogger should be an instance variable of type XLogger.

RMI Logging

For the Service Implementation of the adapter there are two approaches, depending on whether the adapter is stateless or stateful.

For stateless adapters, there is only one object created for all copies of the adapter. Therefore, instance variables are shared by all copies of the adapter.

The service name is not available to the Service Implementation of a stateless adapter from the Gentran Integration Suite infrastructure. The EJB must pass the service name to the Service Implementation of the adapter. The service name should not be used in the constructor when creating an XLogger copy—for example, adapterLogger = new XLogger(classsName). AdapterLogger should be a class variable of type XLogger. In this case, use the logging method that includes the service name.

For stateful adapters, there is an object created for each copy of the adapter, but all the threads of that adapter copy share the same object. Therefore, all threads of an adapter copy share instance variables.

The service name is available to the Service Implementation of a stateful adapter from the Gentran Integration Suite infrastructure by calling getServiceName().

The following is sample code:

```
adapterLogger = new XLogger(classsName, getServiceName());
```

The adapterLogger should be a class variable of type XLogger. The easiest way to do this is to construct a new XLogger if one does not already exist.

XLogger Logging Methods

You can use the XLogger logging methods in the EJB and RMI parts of the adapter. However, the availability of the service name varies.

Use the XLogger methods for logging from adapters. These methods format the class name and the service name provided along with a thread identifier, and add the message to be logged. The result is a log message that includes date/time, class name, service name, thread ID, and message.

The following table describes the XLogger class methods and indicates where they are used:

Method	Used by Stateful RMI and EJBs	Used by Stateless RMI
XLogger(String ClassName)		Constructor
XLogger(String ClassName, String ServiceName)	Constructor	
logError(String ServiceName, String message)		Logging errors
logError(String message)	Logging errors	
logException(String ServiceName, String message, Exception e)		Logging fault exceptions
logException(String message, Exception e)	Logging fault exceptions	
logWarn(String ServiceName, String message)		Logging warning messages
logWarn(String message)	Logging warning messages	
logDebug(String ServiceName, String message)		Logging debug messages
logDebug(String message)	Logging debug messages	
log(String ServiceName, String message)		General logging
log(String message)	General logging	

LogService Logging Methods

The *LogService class* provides the following static logging methods. Use these methods at any time in an EJB or RMI part of an adapter. Because each method implies a different logging threshold, some general usage guidelines are provided.

Each log message generated through LogService has a timestamp and logging level. You must supply the name of the originating class and a message.

The following sample code shows an example:

```
[2001-06-12 12:46:55.381] ALL [SiebelEJBBean] Hello World
```

Timestamp	Logging Level	Originating Class	Message
[2001-06-12 12:46:55.381]	ALL	[SiebelEJBBean]	Hello World

The following table lists the log methods and general guidelines for their formatting:

Note: For the BPML specifications that Gentran Integration Suite accepts, see the Gentran Integration Suite *Business Process Guide*.

Method	Format Guideline	Example
logError()	[Class name] General business error. Specific application error.	[SiebelEJBBean] Error: Adapter unable to process request. Action parameter is null.
logException()	[Class name] General business error. Specific application error. The logException() method requires both a message and an exception. This method logs the exception name and stack trace.	[SiebelEJBBean] Exception: Adapter unable to process Read request. Unable to connect to the file system server.
logWarn()	[Class name] General Warning. Expected results or recommended actions to be taken.	[SiebelEJBBean] Warning: Available disk space is less than 1 GB. Delete unnecessary files.
logDebug()	[Class name] Debug message	[SiebelEJBBean] Entering ProcessData method.
log()	[Class name] Log message	[SiebelEJBBean] Adapter started.

Appendix A

File System Adapter Examples

This appendix contains code examples of a working, fully functional File System adapter. Use these examples as a reference to help you to create your own adapter.

This appendix contains the following sample files:

- FileSystemServer File
- FileSystemServerImpl File
- File System Adapter XML
- FileSystem XML
- Filesystem_en File
- FileSystemImpl File
- FileSystemServer File
- FileSystemServerImpl File
- FSFromCollectInfo File
- FSFromReadFile File
- FSToCollectInfo File
- FSToExtractInfo File
- WFStartThread File

FileSystemServer File

The following sample code shows a FileSystemServer.java file:

```
/**
 * Copyright: Sterling Commerce, 2000-2001. All rights reserved.
 *
 * This software is the proprietary information of
   Sterling Commerce,
 * Inc. Use is subject to license terms.
 */
package com.sterlingcommerce.woodstock.services.filesystem;
```

```
import weblogic.rmi.Remote;
import weblogic.rmi.RemoteException;
import com.sterlingcommerce.woodstock.services.AdapterException;
import com.sterlingcommerce.woodstock.services.IAdapterRMI;

/**
 * Title:       FileSystem Adapter project
 * Description: This adapter can collect or extract files
     to a file system
 * Copyright:   Sterling Commerce, 2000. All rights reserved.
 * @version 1.0, 2/6/2001
 * @since Woodstock 2.0
 */
public interface FileSystemServer extends IAdapterRMI
{
    String[] scanFolder(String folder, String fileFilter, boolean
      useSubFolders) throws AdapterException, RemoteException;

    void createFile(String absoluteFileName) throws
      AdapterException, RemoteException;

    byte[] readFile(String absoluteFileName) throws
      AdapterException, RemoteException;

    void writeFile(String absoluteFileName, byte[] buffer) throws
      AdapterException, RemoteException;

    void deleteFile(String absoluteFileName) throws
      AdapterException, RemoteException;
}
```

FileSystemServerImpl File

The following sample code shows a FileSystemServerImpl.java file:

```
/**
 * Copyright: Sterling Commerce, 2000-2001. All rights reserved.
 *
 * This software is the proprietary information of
   Sterling Commerce,
 * Inc. Use is subject to license terms.
 */
package com.sterlingcommerce.woodstock.services.filesystem;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.util.Vector;
import java.util.*;
import java.util.Properties;
import com.sterlingcommerce.woodstock.util.WildCardFilter;
import com.sterlingcommerce.woodstock.util.frame.Manager;
import com.sterlingcommerce.woodstock.services.*;
import com.sterlingcommerce.woodstock.workflow.*;
import com.sterlingcommerce.woodstock.util.frame.log.*;
```

```
/**
 * Implements the I/O routines for the File System EJB
 * @version 1.0, 2/6/2001
 * @since Woodstock 2.0
 */
public class FileSystemServerImpl extends IAdapterImpl implements FileSystemServer
{
    public FileSystemServerImpl() { super(); }
    public void refreshAdapter(Properties p) {}
    public String message(String s) { return s; }
    public void startupAdapter(Properties p) {}
    public void shutdownAdapter() {}

    /**
     * Method for returning an array of filenames in a directory
     * @param folderName name of the directory
     * @param useSubFolders
     * @since Woodstock 2.0
     */
    public String[] scanFolder(String folderName, String fileFilter, boolean useSubFolders)
    {
        registerThread();
        Vector fileVect = new Vector();
        WildCardFilter filter = new WildCardFilter(fileFilter);
        traverseDir(fileVect, filter, folderName, useSubFolders);
        String[] fileNames = null;
        int vectSize = fileVect.size();
        if ( vectSize > 0 )
        {
            fileNames = new String[vectSize];
            while ( vectSize-- > 0 )
            {
                fileNames[vectSize] = (String)fileVect.elementAt(vectSize);
            }
        }
        unregisterThread();
        return fileNames;
    }

    /**
     * Method for recursively traversing a directory structure
     * @param fileVect Vector object used to collect the recursed information
     * @param folderName name of the directory
     * @param useSubFolders
     * @since Woodstock 2.0
     */
   public void traverseDir(Vector fileVect, WildCardFilter filter, String folderName, boolean
useSubFolders)
    {
        if ( folderName != null )
        {
            File folder = new File(folderName);
            File[] fileList = folder.listFiles(filter);
            if ( fileList != null )
            {
                for ( int i = 0; i < fileList.length; i++ )
                {
                    if ( !fileList[i].isDirectory() )
                    {
                        fileVect.add(fileList[i].getAbsolutePath());
                    }
                    else if ( useSubFolders )
```

```
                    {
                        traverseDir(fileVect, filter, fileList[i].getAbsolutePath(),
useSubFolders);
                    }
                }
            }
        }
    }

    /**
     * Method for creating a file on disk
     * @param absoluteFileName name of the file with its absolute path
     * @exception AdapterException
     * @since Woodstock 2.0
     */
    public void createFile(String absoluteFileName) throws AdapterException
    {
        registerThread();
        try
        {
            new File(absoluteFileName);
        }
        catch(Exception e)
        {
            unregisterThread();
            throw new AdapterException(e);
        }
        unregisterThread();
    }

    /**
     * Method for deleting a file on disk
     * @param absoluteFileName name of the file with its absolute path
     * @exception AdapterException
     * @since Woodstock 2.0
     */
     public void deleteFile(String absoluteFileName) throws AdapterException
     {
        registerThread();
        File file = null;
        try
        {
            file = new File(absoluteFileName);
            file.delete();
        }
        catch(Exception e)
        {
            unregisterThread();
            throw new AdapterException(e);
        }
        unregisterThread();
     }

    /**
     * Method for reading a file from disk.<p>
     * @param absoluteFileName name of the file (inc. its absolute path name)
     * @return byte[] array of bytes making up the data
     * @exception AdapterException
     * @since Woodstock 2.0
     */
    public byte[] readFile(String absoluteFileName) throws AdapterException
    {
```

```
        registerThread();
        BufferedInputStream biStream = null;
        File file = new File(absoluteFileName);
        long fileSize = 0;
        int cbRead = 0;
        byte[] buffer = null;

        try
        {
            biStream = new BufferedInputStream(new FileInputStream(file));
            fileSize = file.length();
            buffer = new byte[(int)fileSize];
            cbRead = biStream.read(buffer, 0, (int)fileSize);
        }
        catch(Exception e)
        {
            unregisterThread();
            throw new AdapterException(e);
        }
        finally
        {
            if ( biStream != null )
            {
                try
                {
                    biStream.close();
                }
                catch(Exception e)
                {
                    LogService.out.logWarn(e.getMessage());
                }
            }
        }

        unregisterThread();
        if ( cbRead > 0 )
        {
            return buffer;
        }
        else
        {
            return null;
        }
    }

    //JOE - STUFF
    private static Hashtable threadTable = new Hashtable();
    private final static int thresh = 10; // only do 10 threads

    public void workflowStart(String wfId, String name, byte[] stuff, String path, String
svcName) throws AdapterException
    {
        synchronized(threadTable)
        {
            registerThread();
            try
            {
                int iwfId = -1;
                if ( wfId != null && !wfId.equals("") )
                {
                    try
                    {
```

```
                        iwfId = Integer.parseInt(wfId);
                    }
                    catch(NumberFormatException nfe)
                    {
                        iwfId = -1;
                    }
                }
                InitialWorkFlowContext iwfc = new InitialWorkFlowContext();
                iwfc.setWorkFlowDefId(iwfId);
                iwfc.setDocumentName(name);
                iwfc.setDocumentBody(stuff);

                if ( threadTable.size() > thresh )
                {
                    waitForComplete();
                }

                WFStartThread wfst = new WFStartThread(iwfc, path, svcName);
                Thread t = new Thread(wfst);
               String id = "FS." + wfId + "." + name + "." + System.currentTimeMillis() + ":"
+ iwfc.hashCode();
                threadTable.put(id, t);
                t.start();
            }
            catch(Exception e)
            {
                unregisterThread();
                throw new AdapterException(e);
            }
            unregisterThread();
        }
    }

    private void waitForComplete()
    {
        Thread t = null;
        Enumeration e = null;
        boolean again = true;
        String k;
        int ct = 0;
        while ( again && ct < 100000 )
        {
            e = threadTable.keys();
            while ( e.hasMoreElements() )
            {
                k = (String)e.nextElement();
                t = (Thread)threadTable.get(k);
                if ( !t.isAlive() )
                {
                    threadTable.remove(k);
                    again = false;
                }
            }
            ct++;
            try
            {
                Thread.sleep(100);
            }
            catch(Exception ex)
            {
                ex.printStackTrace();
            }
```

```
        }
    }

    /**
     * Method for writing a file to disk.<p>
     * @param absoluteFileName name of the file with its absolute path
     * @param buffer array of byte to write to the file
     * @exception AdapterException
     * @since Woodstock 2.0
     */
    public void writeFile(String absoluteFileName, byte[] buffer) throws AdapterException
    {
        registerThread();
        BufferedOutputStream boStream = null;
        try
        {
            boStream = new BufferedOutputStream(new FileOutputStream(new
File(absoluteFileName)));
            boStream.write(buffer, 0, buffer.length);
        }
        catch(Exception e)
        {
            unregisterThread();
            throw new AdapterException(e);
        }
        finally
        {
            if ( boStream != null )
            {
                try
                {
                    boStream.flush();
                    boStream.close();
                }
                catch(Exception e)
                {
                    LogService.out.logWarn(e.getMessage());
                }
            }
        }
        unregisterThread();
    }
}
```

File System Adapter XML

The following sample code shows a File System adapter XML:

```
<SERVICES>

<SERVICE name="FileSystem" description="fs.description" label="fs.label"
implementationType="EJB" JNDIName="FileSystemEJBHome" type="Adapter" adapterType="STATELESS"
adapterClass="com.sterlingcommerce.woodstock.services.filesystem.FileSystemServerImpl"
version="1.0" SystemService="NO">
 <BP_XML>
    <![CDATA[<process name="Scheduler_&service_name;">
       <sequence>
          <operation name="Service">
```

```
            <participant name="&service_name;"/>
            <output message="Xout">
               <assign to="." from="*"></assign>
               <assign to="Action">FS_COLLECT</assign>
            </output>
            <input message="Xin">
               <assign to="." from="*"></assign>
            </input>
         </operation>
      </sequence>
   </process>]]>
 </BP_XML>
 <VARS type="wfd">
    <GROUP title="fs.wfd.group1.title" instructions="fs.wfd.group1.instructions">
       <VARDEF varname="Action" type="String" htmlType="select" validator="ALPHANUMERIC"
label="fs.action" options="fstype" />
    </GROUP>
  </VARS>
  <VARS type="instance">
    <GROUP title="fs.instance.group1.title" instructions="fs.instance.group1.instructions">
      <VARDEF varname="collectionFolder" type="String" htmlType="text"
validator="ALPHANUMERIC" size="30" maxsize="250" label="fs.cfolder" />
     <VARDEF varname="useSubFolder" type="String" htmlType="radio" validator="ALPHANUMERIC"
options="radio2" label="fs.subfolders" />
    </GROUP>
    <GROUP title="fs.instance.group2.title" instructions="fs.instance.group2.instructions">
      <VARDEF varname="extractionFolder" type="String" htmlType="text"
validator="ALPHANUMERIC" size="30" maxsize="250" label="fs.efolder" />
     <VARDEF varname="assignFilename" type="String" htmlType="radio" validator="ALPHANUMERIC"
options="fsfilename" label="fs.filename">
         <SUBGROUP dependencyvalue="true" title="fs.instance.group2a.title"
instructions="fs.instance.group2a.instructions">
            <VARDEF varname="assignedFilename" type="String" htmlType="text"
validator="ALPHANUMERIC" size="40" maxsize="250" label="fs.extractfilename" />
         </SUBGROUP>
      </VARDEF>
    </GROUP>
    <GROUP title="system.sched.title" instructions="system.sched.instructions">
     <VARDEF varname="schedDay" type="String" htmlType="select" validator="NUMBER"
label="system.schedDay.label" options="schedDay"/>
     <VARDEF varname="schedHour" type="String" htmlType="select" validator="NUMBER"
label="system.schedHour.label" options="schedHour"/>
     <VARDEF varname="schedMinute" type="String" htmlType="select" validator="NUMBER"
label="system.schedMinute.label" options="schedMinute"/>
     <VARDEF varname="schedOnMin" type="String" htmlType="select" validator="ALPHANUMERIC"
label="system.schedOnMin.label" options="schedOnMinute"/>
     <VARDEF varname="schedMerid" type="String" htmlType="select" validator="NUMBER"
label="system.schedMerid.label" options="schedMeridian"/>
    </GROUP>
  </VARS>
  <VARS type="assignbp">
    <GROUP title="bpsched.assignbp.title" instructions="bpsched.assignbp.instructions">
      <VARDEF varname="initialWorkFlowId" type="String" htmlType="select"
validator="ALPHANUMERIC" label="bpsched.assignbp.title" options="bplist" />
    </GROUP>
  </VARS>
</SERVICE>

<OPTION name="fstype">
    <ELE value="FS_COLLECT" displayname="fs.Collection" />
    <ELE value="FS_EXTRACT" displayname="fs.Extraction" />
</OPTION>
```

```
<OPTION name="fsfilename">
    <ELE value="false" displayname="fs.useOriginal" />
    <ELE value="true" displayname="fs.assignName" />
</OPTION>

</SERVICES>
```

FileSystem XML

The following sample code is a FileSystem XML file:

```
-->
- <SERVICES>
- <SERVICE name="FileSystem" description="fs.description" label="fs.label"
implementationType="CLASS"
JNDIName="com.sterlingcommerce.woodstock.services.filesystem.FileSystemImpl" type="Adapter"
adapterType="STATELESS"
adapterClass="com.sterlingcommerce.woodstock.services.filesystem.FileSystemServerImpl"
version="1.0" SystemService="NO">
- <BP_XML>
- <![CDATA[
<process name="Scheduler_&service_name;">
       <sequence>
          <operation name="Service">
             <participant name="&service_name;"/>
             <output message="Xout">
                <assign to="." from="*"></assign>
                <assign to="Action">FS_COLLECT</assign>
             </output>
             <input message="Xin">
                <assign to="." from="*"></assign>
             </input>
          </operation>
       </sequence>
    </process>
]]>
</BP_XML>
- <VARS type="wfd">
- <GROUP title="fs.wfd.group1.title" instructions="fs.wfd.group1.instructions">
<VARDEF varname="Action" type="String" htmlType="select" validator="ALPHANUMERIC"
label="fs.action" options="fstype" />
<VARDEF varname="appendOnExtract" type="String" htmlType="select" validator="ALPHANUMERIC"
label="fs.append" options="radio2" />
<VARDEF varname="deleteAfterCollect" type="String" htmlType="select" validator="ALPHANUMERIC"
label="fs.delete" options="radio2" />
<VARDEF varname="collectZeroByteFiles" type="String" htmlType="select"
validator="ALPHANUMERIC" label="fs.zero" options="radio2" />
<VARDEF varname="fileModTimeThreshold" type="String" htmlType="text" validator="ALPHANUMERIC"
size="30" maxsize="250" label="fs.modtime" />
<VARDEF varname="maxThreads" type="String" htmlType="text" validator="NUMBER" size="5"
maxsize="5" label="fs.maxthreads" />
</GROUP>
</VARS>
- <VARS type="instance">
- <GROUP title="fs.instance.group1.title" instructions="fs.instance.group1.instructions">
```

```
<VARDEF varname="collectionFolder" type="String" htmlType="text" validator="ALPHANUMERIC"
size="30" maxsize="250" label="fs.cfolder" />
<VARDEF varname="filter" type="String" htmlType="text" size="30" maxsize="250"
label="fs.filter" required="NO" />
<VARDEF varname="useSubFolders" type="String" htmlType="radio" validator="ALPHANUMERIC"
options="radio2" label="fs.subfolders" />
<VARDEF varname="keepPath" type="String" htmlType="radio" validator="ALPHANUMERIC"
options="radio2" label="fs.keepPath" defaultVal="false" />
- <VARDEF varname="bootstrap" type="String" htmlType="radio" validator="ALPHANUMERIC"
options="radio2" label="fs.bootstrap">
- <SUBGROUP dependencyvar="bootstrap" dependencyvalue="true" title="bpsched.assignbp.title"
instructions="bpsched.assignbp.instructions">
<VARDEF varname="initialWorkFlowId" type="String" htmlType="select" validator="ALPHANUMERIC"
label="bpsched.assignbp.title" options="bplist" />
</SUBGROUP>
<SUBGROUP dependencyvar="bootstrap" dependencyvalue="true"
handler="com.sterlingcommerce.woodstock.ui.ScheduleConfig" wizard="Scheduler" />
</VARDEF>
</GROUP>
- <GROUP title="fs.instance.group2.title" instructions="fs.instance.group2.instructions">
<VARDEF varname="extractionFolder" type="String" htmlType="text" validator="ALPHANUMERIC"
size="30" maxsize="250" label="fs.efolder" />
- <VARDEF varname="assignFilename" type="String" htmlType="radio" validator="ALPHANUMERIC"
options="fsfilename" label="fs.filename">
- <SUBGROUP dependencyvalue="true" title="fs.instance.group2a.title"
instructions="fs.instance.group2a.instructions">
<VARDEF varname="assignedFilename" type="String" htmlType="text" size="40" maxsize="250"
label="fs.extractfilename" />
</SUBGROUP>
</VARDEF>
</GROUP>
</VARS>
</SERVICE>
- <OPTION name="fstype">
<ELE value="FS_COLLECT" displayname="fs.Collection" />
<ELE value="FS_EXTRACT" displayname="fs.Extraction" />
</OPTION>
- <OPTION name="fsfilename">
<ELE value="false" displayname="fs.useOriginal" />
<ELE value="true" displayname="fs.assignName" />
</OPTION>
</SERVICES>
```

Filesystem_en File

The following sample code is a Filesystem_en file:

```
fs.label = File System Adapter
fs.description = Collects and Extracts files from a file system.
as2fs.label = AS2 File System Adapter
as2fs.description = Collects and Extracts as2 files from a file system.

fs.action = Action
fs.append = Append to file when extracting?
fs.delete = Delete file after collecting?
fs.zero = Collect zero byte files?
fs.modtime = The modification time of a file must be older than the number of seconds specified
or it will not be collected.
```

```
fs.cfolder = Collection folder
fs.efolder = Extraction folder
fs.filter = Filename filter
fs.pollinterval = Poll Interval (mins)
fs.subfolders = Collect files from sub folders within and including the collection folder?
fs.keepPath = Use the absolute file path name for the document name?
fs.bootstrap = Start a business process once files are collected?
fs.extractfilename = Filename
fs.filename = Filenaming convention
fs.Collection = Collection
fs.Extraction = Extraction
fs.useOriginal = Use the original filename as the extracted filename
fs.assignName = Assign a specific name
fs.maxthreads = Maximum number of threads to use when bootstrapping collected files
fs.contract = Contratc

fs.wfd.group1.title = Workflow Properties
fs.wfd.group1.instructions = Specify the appropriate workflow settings.

fs.instance.group1.title = Collection
fs.instance.group1.instructions = \
<table border=0 cellpadding=1 cellspacing=1>\
<tr>\
  <td colspan=2 valign="top" align="left" class='info'>\
    Specify the appropriate settings for collecting data.\
  </td>\
</tr>\
<tr>\
  <TD class='bullet'>&#149;</TD>\
  <td valign="top" align="left" class='info'>\
    Specify the folder to collect files from.\
  </td>\
</tr>\
<tr>\
  <TD class='bullet'>&#149;</TD>\
  <td valign="top" align="left" class='info'>\
    An optional filter can be used to only collect certain files (i.e. *.txt).\
  </td>\
</tr>\
<tr>\
  <TD class='bullet'>&#149;</TD>\
  <td valign="top" align="left" class='info'>\
    Specify whether to search any sub folders within the main collection folder.\
  </td>\
</tr>\
<tr>\
  <TD class='bullet'>&#149;</TD>\
  <td valign="top" align="left" class='info'>\
    Specify whether or not a business process is started once files are collected.\
  </td>\
</tr>\
</table>

fs.instance.group2.title = Extraction
fs.instance.group2.instructions = \
<table border=0 cellpadding=1 cellspacing=1>\
<tr>\
  <td colspan=2 valign="top" align="left" class='info'>\
    Specify the appropriate settings for extracting files.\
  </td>\
</tr>\
<tr>\
```

```
  <TD class='bullet'>&#149;</TD>\
  <td valign="top" align="left" class='info'>\
    Specify the folder to extract files to.\
  </td>\
</tr>\
<tr>\
  <TD class='bullet'>&#149;</TD>\
  <td valign="top" align="left" class='info'>\
    Specify whether to use the original document filename or assign a different one.\
  </td>\
</tr>\
</table>

fs.instance.group2a.title = User defined
fs.instance.group2a.instructions = \
<table border=0 cellpadding=1 cellspacing=1>\
<tr>\
  <td colspan=2 valign="top" align="left" class='info'>\
    Specify the filename to assign to the extracted files.  To generate unique filenames, the
escape code %^ will be replaced with a unique number in the format yyyymmddhhmmsslll (i.e.
specifying Rcv%^.dat will generate Rcv200110191259123.dat).\
  </td>\
</tr>\
</table>
```

FileSystemImpl File

The following sample code is a FileSystemImpl file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.io.File;
import java.util.Arrays;
import com.sterlingcommerce.woodstock.util.Util;
import com.sterlingcommerce.woodstock.workflow.WorkFlowDef;
import com.sterlingcommerce.woodstock.workflow.WorkFlowContext;
import com.sterlingcommerce.woodstock.workflow.WorkFlowException;
import com.sterlingcommerce.woodstock.services.IService;
import com.sterlingcommerce.woodstock.services.XLogger;
import com.sterlingcommerce.woodstock.util.frame.Manager;
import com.sterlingcommerce.woodstock.util.frame.log.LogService;
import com.sterlingcommerce.woodstock.util.frame.lock.LockManager;
import com.sterlingcommerce.woodstock.services.controller.ServicesControllerImpl;
import org.w3c.dom.Document;
import org.w3c.dom.Node;

public class FileSystemImpl implements IService {
    static final int DEF_LOCKLOOP = 5000;
    static final int DEF_AGELOOP = 6;


//#############################################################################################
###
    public WorkFlowContext processData(WorkFlowContext wfc) throws WorkFlowException {
        String badCfg = "No 'Action' or request specified";
        String svcName = wfc.getServiceName();
```

```
        XLogger log = new XLogger("FileSystemImpl", svcName);
        ServicesControllerImpl sci = ServicesControllerImpl.getInstance();
        sci.harnessRegister(String.valueOf(wfc.getWorkFlowId()), svcName);
        try {
            FileSystemServer rmi = (FileSystemServer)sci.getAdapter(svcName);
            if ( rmi == null ) {
                handleError(wfc, log, "RMI instance is null");
            }
            else {
                wfc.suspendTransaction();
                String action = wfc.getParm("Action"); // determine action
                if ( action == null ) {
                    Node node = (Node)wfc.getWFContent("/*");
                    if ( node != null ) {
                        String inputMsg = node.getNodeName();
                        if ( inputMsg != null ) {
                            if ( inputMsg.equals("importFileRequest") ) {
                                doImportFile(wfc, log, rmi);
                            }
                            else if ( inputMsg.equals("exportDocumentRequest") ) {
                                doExportFile(wfc, log, rmi);
                            }
                            else { // no idea what to do
                                handleError(wfc, log, badCfg);
                            }
                        }
                        else { // inputMsg is null
                            handleError(wfc, log, badCfg);
                        }
                    }
                    else {  // node is null
                        handleError(wfc, log, badCfg);
                    }
                }
                else if ( action.equals("FS_EXTRACT") ) {
                    doExtract(wfc, log, rmi);
                }
                else if ( action.equals("FS_COLLECT") ) {
                    doCollect(wfc, log, rmi, svcName);
                }
                else {
                    handleError(wfc, log, "Unknown 'Action' value: " + action);
                }
                wfc.resumeTransaction();
            }
        }
        catch(Exception e) {
            handleException(wfc, log, "Exception in processData", e);
        }
        sci.unregisterThread();
        return wfc;
    }


//#############################################################################################
###
    private void doExtract(WorkFlowContext wfc, XLogger log, FileSystemServer rmi) {
        String msg = "Exception in doExtract"; // a multi-use message string var
        try {
            String advStatus = null;
            FSToExtractInfo tei = new FSToExtractInfo();
            tei.folder = wfc.getParm("extractionFolder");
```

```
            if ( tei.folder == null || tei.folder.length() == 0 ) {
                handleError(wfc, log, "Missing required 'extractionFolder' parameter");
            }
            else {
                com.sterlingcommerce.woodstock.workflow.Document doc =
wfc.getPrimaryDocument();
                if ( doc == null ) {
                    handleError(wfc, log, "Primary document is null");
                }
                else {
                    tei.data = doc.getBody();
                    if ( tei.data == null ) {
                        advStatus = "0 byte file";
                    }
                    tei.bodyName = doc.getBodyName();
                    if ( tei.bodyName == null || tei.bodyName.length() == 0 ) {
                        tei.bodyName = Util.getUniqueFileName("%^.dat");
                    }
                    if ( "true".equals(wfc.getParm("assignFilename")) ) {
                        tei.assignedFilename = wfc.getParm("assignedFilename");
                    }
                    if ( "true".equals(wfc.getParm("appendOnExtract")) ) {
                        tei.append = true;
                    }
                    msg = "Exception in writeFile"; // load in case it throws exception
                    rmi.writeFile(tei);
                    setStatus(wfc, WorkFlowContext.SUCCESS, advStatus);
                }
            }
        }
        catch(Exception e) {
            handleException(wfc, log, msg, e);
        }
    }


//#############################################################################################
###
    private void doCollect(WorkFlowContext wfc, XLogger log, FileSystemServer rmi, String
svcName) {
        try {
            FSToCollectInfo tci = new FSToCollectInfo();
            tci.wfctm = wfc.packMessageToChild();
            tci.svcName = svcName;
            String tmp = wfc.getParm("fileModTimeThreshold");
            if ( tmp == null || tmp.length() == 0 ) {
                tmp = Manager.getProperty("FSAdapterFileModSeconds");
            }
            try {
                tci.modTime = Integer.parseInt(tmp);
            }
            catch(Exception e) {
                tci.modTime = 30;
            }
            tci.modTime *= 1000; // adjust to milliseconds

            tci.folder = wfc.getParm("collectionFolder");
            if ( tci.folder == null || tci.folder.length() == 0 ) {
                handleError(wfc, log, "Missing required 'collectionFolder' parameter");
            }
            else {
                tmp = wfc.getParm("useSubFolders");
```

```
                if ( "true".equals(tmp) ) {
                    tci.useSubFolders = true;
                }
                tmp = wfc.getParm("keepPath");
                if ( "true".equals(tmp) ) {
                    tci.keepPath = true;
                }
                tmp = wfc.getParm("deleteAfterCollect");
                if ( "false".equals(tmp) ) {
                    tci.delete = false;
                }
                tmp = wfc.getParm("collectZeroByteFiles");
                if ( "true".equals(tmp) ) {
                    tci.getzero = true;
                }
                tci.filter = wfc.getParm("filter");
                if ( tci.filter == null || tci.filter.length() == 0 ) {
                    tci.filter = "*";
                }
                tmp = wfc.getParm("bootstrap");
                if ( "false".equals(tmp) ) {
                    doCollectNoBootstrap(wfc, log, rmi, tci);
                }
                else {
                    doCollectWithBootstrap(wfc, log, rmi, tci);
                }
            }
        }
        catch(Exception e) {
            handleException(wfc, log, "Exception in doCollect", e);
        }
    }


//#############################################################################################
###
    private void doCollectWithBootstrap(WorkFlowContext wfc, XLogger log, FileSystemServer
rmi, FSToCollectInfo tci) {
        String msg = "Exception in doCollectWithBootstrap";
        boolean isLocked = false;
        try {
            String iwfId = wfc.getParm("initialWorkFlowId");
            String iwfName = wfc.getParm("initialWorkFlowName");
            if ( iwfId == null ) {
                if ( iwfName != null ) {
                    try {
                        tci.iwfid = WorkFlowDef.getIDForName(iwfName);
                    }
                    catch(Exception e) {
                        tci.iwfid = -1;
                    }
                }
            }
            else {
                try {
                    tci.iwfid = WorkFlowDef.getIDForName(iwfId);
                }
                catch(Exception e) {
                    try {
                        tci.iwfid = Integer.parseInt(iwfId);
                    }
                    catch(NumberFormatException nfe) {
```

```
                        tci.iwfid = -1;
                    }
                }
            }
            if ( tci.iwfid == -1 ) {
                handleError(wfc, log, "Required 'initialWorkFlowId' parameter not found or
invalid");
            }
            else {
                try {
                    tci.maxThreads = Integer.parseInt(wfc.getParm("maxThreads"));
                }
                catch(Exception e) {
                    tci.maxThreads = 10;
                }
                if ( tci.maxThreads < 1 ) {
                    tci.maxThreads = 1; // gotta have at least one
                }
                if ( LockManager.isLocked(tci.folder) ) {
                    setStatus(wfc, WorkFlowContext.SUCCESS, tci.folder + " is locked");
                }
                else {
                    LockManager.lock(tci.folder, tci.svcName, 0, true);
                    isLocked = true;
                    msg = "Exception in collect";
                    FSFromCollectInfo fci = rmi.collect(tci);
                    if ( fci.workflows != null && fci.filesCollected > 0 ) {
                        wfc.addBootStrapWorkFlows(fci.workflows);
                    }
                    LockManager.unlock(tci.folder);
                    isLocked = false;
                    if ( fci.couldntRead ) {
                        if ( fci.filesCollected == 0 ) {
                            handleError(wfc, log, "Unable to collect any files");
                        }
                        else {
                            setStatus(wfc, WorkFlowContext.SUCCESS, "Unable to collect one or
more files");
                        }
                    }
                    else if ( fci.filesCollected == 0 ) {
                        setStatus(wfc, WorkFlowContext.SUCCESS, "No files to collect");
                    }
                    else {
                        setStatus(wfc, WorkFlowContext.SUCCESS, fci.filesCollected + " files
collected");
                    }
                }
            }
        }
        catch(Exception e) {
            if ( isLocked ) {
                LockManager.unlock(tci.folder);
            }
            handleException(wfc, log, msg, e);
        }
    }


//#############################################################################################
###
```

```
    private void doCollectNoBootstrap(WorkFlowContext wfc, XLogger log, FileSystemServer rmi,
FSToCollectInfo tci) {
        String msg = "Exception in doCollectNoBootstrap";
        boolean isLocked = false;
        try {
            if ( tci.filter.indexOf("*") != -1 ) {
                int lockLoop = 5;
                while ( lockLoop-- > 0 ) {
                    if ( LockManager.isLocked(tci.folder) ) {
                        if ( lockLoop == 0 ) {
                            handleError(wfc, log, tci.folder + " is still locked after 5
attempts");
                            return;
                        }
                        try {
                            Thread.sleep(DEF_LOCKLOOP);
                        }
                        catch(Exception e) {
                            log.logException("Exception during Thread.sleep in lock loop", e);
                        }
                    }
                    else {
                        break;
                    }
                }
                LockManager.lock(tci.folder, tci.svcName, 0, true);
                isLocked = true;
            }
            int filesRead = 0; // used when all reads return null so we can set "no files to
collect"
            boolean couldntRead = false;
            msg = "Exception in scanFolder";
           String[] files = rmi.scanFolder(tci.folder, tci.filter, tci.useSubFolders, true);
// get list of files
            if ( files != null ) {
                FSFromReadFile frf = null;
                int i = 0;
                int aged = DEF_AGELOOP;
                while ( i < files.length ) {
                    msg = "Exception in readFile";
                    frf = rmi.readFile(files[i], tci.modTime, true);
                    if ( frf.goodfile ) {
                        if ( frf.data == null ) {
                            if ( tci.getzero ) {
                                frf.data = "".getBytes(); // allows for collecting zero byte
files
                            }
                            else {
                                i++;
                                continue;
                            }
                        }
                        filesRead++;
                        com.sterlingcommerce.woodstock.workflow.Document doc =
wfc.createDocument();
                        doc.setBody(frf.data);
                        String tmp = files[i];
                        if ( !tci.keepPath ) {
                            tmp = new File(files[i]).getName();
                        }
                        doc.setBodyName(tmp);
                        wfc.putPrimaryDocument(doc);
```

```
                        if ( tci.delete ) {
                            msg = "Exception in deleteFile";
                            rmi.deleteFile(files[i]);
                        }
                        break;
                    }
                    else if ( frf.data != null && Arrays.equals(frf.data,
FileSystemServerImpl.CANTREAD) ) {
                        couldntRead = true;
                    }
                    else if ( files.length == 1 && aged-- > 0 ) {
                        // if we get to this point, we're trying to read just one file and it
wasn't aged enough
                        try {
                            Thread.sleep(tci.modTime);
                        }
                        catch(Exception e) {
                            log.logException("Exception during Thread.sleep", e);
                        }
                       continue; // won't increment so it will try getting the same file again
                    }
                    i++;
                }
            }
            if ( couldntRead ) {
                if ( filesRead == 0 ) {
                    handleError(wfc, log, "Unable to collect any files");
                }
                else {
                    setStatus(wfc, WorkFlowContext.SUCCESS, null);
                }
            }
            else if ( filesRead == 0 ) {
                handleError(wfc, log, "No files to collect");
            }
            else {
                setStatus(wfc, WorkFlowContext.SUCCESS, null);
            }
        }
        catch(Exception e) {
            handleException(wfc, log, msg, e);
        }
        finally {
            if ( isLocked ) {
                LockManager.unlock(tci.folder);
            }
        }
    }


//#############################################################################################
###
    private void doImportFile(WorkFlowContext wfc, XLogger log, FileSystemServer rmi) {
        String msg = "Exception in doImportFile";
        try {
            if ( LogService.out.debug ) {
                log.logDebug("importFileRequest");
            }
            String filename = wfc.getParm("filename"); // ImportFile
            if ( filename == null || filename.length() == 0 ) {
                handleError(wfc, log, "Missing required 'filename' parameter");
            }
```

```
            else {
                String dirname = wfc.getParm("dirname");
                FSFromReadFile frf = null;
                int aged = DEF_AGELOOP;
                while ( aged-- > 0 ) {
                    msg = "Exception in importFile";
                    frf = rmi.importFile(dirname, filename);
                    if ( frf.goodfile ) {
                        if ( frf.data == null ) {
                            frf.data = "".getBytes(); // allows for collecting zero byte files
                        }
                        com.sterlingcommerce.woodstock.workflow.Document doc =
wfc.createDocument();
                        doc.setBody(frf.data);
                        doc.setBodyName(filename);
                        wfc.putDocument("document", doc);
                        Object value = doc.getDocumentId();
                        if ( value instanceof org.w3c.dom.Document ) {
                            value = ((org.w3c.dom.Document)value).getDocumentElement();
                        }
                        wfc.setWFContent("doc:document-id", value, false);
                        setStatus(wfc, WorkFlowContext.SUCCESS, null);
                        aged = 1; // just in case it's zero at the time it works
                        break;
                    }
                    else if ( frf.data != null && Arrays.equals(frf.data,
FileSystemServerImpl.CANTREAD) ) {
                        handleError(wfc, log, "Unable to read " + frf.importFile);
                    }
                    else {
                        // if we get here, the file is probably not aged enough
                        try {
                            Thread.sleep(FileSystemServerImpl.DEF_MODTIME);
                        }
                        catch(Exception e) {
                            log.logException("Exception during Thread.sleep", e);
                        }
                    }
                }
                if ( aged == 0 ) { // if we get here - importFileRequest did not work
                    handleError(wfc, log, frf.importFile + " was not collected");
                }
            }
        }
        catch(Exception e) {
            handleException(wfc, log, msg, e);
        }
    }


//#############################################################################################
###
    private void doExportFile(WorkFlowContext wfc, XLogger log, FileSystemServer rmi) {
        String msg = "Exception in doExportFile";
        try {
            if ( LogService.out.debug ) {
                log.logDebug("exportDocumentRequest");
            }
            String docId = wfc.getParm("doc:document-id");
            String filename = wfc.getParm("filename");
            String prefix = wfc.getParm("filename-prefix");
            if ( docId == null || docId.trim().length() == 0 ) {
```

```
                handleError(wfc, log, "Missing required 'doc:document-id' parameter");
            }
            else {
                if ( prefix == null || prefix.trim().length() == 0 ) {
                    prefix = "document-";
                }
                if ( filename == null || filename.trim().length() == 0 ) {
                    filename = prefix + docId;
                }
                com.sterlingcommerce.woodstock.workflow.Document doc = new
com.sterlingcommerce.woodstock.workflow.Document(docId);
                if ( doc == null ) {
                    handleError(wfc, log, "Document is null: " + docId);
                }
                else {
                    msg = "Exception in exportFile";
                    rmi.exportFile(filename, doc.getBody());
                    wfc.setWFContent("filename", filename);
                    setStatus(wfc, WorkFlowContext.SUCCESS, null);
                }
            }
        }
        catch(Exception e) {
            handleException(wfc, log, msg, e);
        }
    }


//#############################################################################################
###
    private void handleError(WorkFlowContext wfc, XLogger log, String advStatus) {
        log.logError(advStatus);
        setStatus(wfc, WorkFlowContext.ERROR, advStatus);
    }


//#############################################################################################
###
   private void handleException(WorkFlowContext wfc, XLogger log, String advStatus, Exception
e) {
        log.logException(advStatus, e);
        setStatus(wfc, WorkFlowContext.ERROR, e.getMessage());
    }


//#############################################################################################
###
    private void setStatus(WorkFlowContext wfc, int basicStatus, String advStatus) {
        wfc.setBasicStatus(basicStatus);
        if ( advStatus != null ) {
            wfc.setAdvancedStatus(advStatus);
```

FileSystemServer File

The following sample code is a FileSystemServer file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.rmi.*;
import java.rmi.server.*;
import java.rmi.RemoteException;
import com.sterlingcommerce.woodstock.services.AdapterException;
import com.sterlingcommerce.woodstock.services.IAdapterRMI;

/**
 * Remote interface of the I/O routines used by the File System Adapter
 * @since Woodstock 2.0
 */
public interface FileSystemServer extends IAdapterRMI
{
    public FSFromCollectInfo collect(FSToCollectInfo tci) throws AdapterException,
RemoteException;
    public String[] scanFolder(String folder, String fileFilter, boolean useSubFolders,
boolean doreg) throws AdapterException, RemoteException;
    public FSFromReadFile readFile(String absoluteFileName, int modTime, boolean doreg) throws
AdapterException, RemoteException;
    public FSFromReadFile importFile(String dirname, String filename) throws AdapterException,
RemoteException;
    public void exportFile(String filename, byte[] data) throws AdapterException,
RemoteException;
    public void writeFile(FSToExtractInfo tei) throws AdapterException, RemoteException;
    public void deleteFile(String absoluteFileName) throws AdapterException, RemoteException;
}
```

FileSystemServerImpl File

The following sample code is a FileSystemServerImpl file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.ByteArrayOutputStream;
import java.util.Properties;
import java.util.Vector;
import com.sterlingcommerce.woodstock.util.Util;
import com.sterlingcommerce.woodstock.util.WildCardFilter;
import com.sterlingcommerce.woodstock.services.AdapterException;
import com.sterlingcommerce.woodstock.services.IAdapterImpl;
import com.sterlingcommerce.woodstock.util.frame.Manager;
import com.sterlingcommerce.woodstock.util.frame.log.LogService;

public class FileSystemServerImpl extends IAdapterImpl implements FileSystemServer {
```

```
    public FileSystemServerImpl() { super(); }
    public void refreshAdapter(Properties p) {}
    public String message(String s) { return s; }
    public void startupAdapter(Properties p) {}
    public void shutdownAdapter() {}
    public static final byte[] CANTREAD = "<Can't @#$ read>".getBytes(); // special chars used
to make it unique (encoding doesn't matter)
    public static final int DEF_MODTIME = 30000; // 30 seconds
    private static final int BUFFER_SIZE = 1024;

    /**
     * Method for collecting files and starting workflows
     * @param tci a FSToCollectInfo class containing all the necessary parms
     * @return a FSFromCollectInfo class containing all the necessary info
     * @since Woodstock 2.0
     */
    public FSFromCollectInfo collect(FSToCollectInfo tci) throws AdapterException {
        registerThread();
        FSFromCollectInfo fci = new FSFromCollectInfo();
        WFStartThread wfst = null;
        Thread t = null;
        try {
            String[] files = scanFolder(tci.folder, tci.filter, tci.useSubFolders, false);
            if ( files != null ) {
                fci.workflows = new Vector(files.length);
                for ( int i = 0; i < files.length; i++ ) {
                    while ( fci.threadCnt == tci.maxThreads ) {
                        Thread.yield();
                    }
                    wfst = new WFStartThread(files[i], tci, fci, this);
                    t = new Thread(wfst);
                    fci.threadCounter(true); // synchronized increment
                    t.start();
                }
                while ( fci.threadCnt > 0 ) {
                    Thread.yield();
                }
            }
        }
        catch(Exception e) {
            LogService.out.logException("Exception in collect method", e);
            unregisterThread();
            throw new AdapterException(e);
        }
        unregisterThread();
        return fci;
    }

    /**
     * Method for returning an array of filenames in a directory
     * @param folderName name of the directory
     * @param fileFilter file filter to apply
     * @param useSubFolders whether to scan subdirectories or not
     * @param doreg controls whether it was called from the collect method internally or from
a RMI call
     * @return A string array of filenames
     * @since Woodstock 2.0
     */
    public String[] scanFolder(String folderName, String fileFilter, boolean useSubFolders,
boolean doreg) throws AdapterException {
        if ( doreg ) {
            registerThread();
```

```
        }
        String folder = null;
        try {
            folder = new File(folderName).getCanonicalPath();
        }
        catch(Exception e) {
            folder = folderName; // should never happen but in case it does, just use it as is
        }
        File wdir = new File(folder);
        if ( !wdir.exists() ) {
            throw new AdapterException(folder + " does not exist");
        }

        Vector fileVect = new Vector(10);
        WildCardFilter filter = new WildCardFilter(fileFilter);
        traverseDir(fileVect, filter, folder, useSubFolders);
        String[] fileNames = null;
        int vectSize = fileVect.size();
        if ( vectSize > 0 ) {
            fileNames = new String[vectSize];
            while ( vectSize-- > 0 ) {
                fileNames[vectSize] = (String)fileVect.elementAt(vectSize);
            }
        }
        if ( doreg ) {
            unregisterThread();
        }
        return fileNames;
    }

    /**
     * Method for recursively traversing a directory structure
     * @param fileVect Vector object used to collect the recursed information
     * @param filter file filter to apply
     * @param folderName name of the directory
     * @param useSubFolders whether to scan subdirectories or not
     * @since Woodstock 2.0
     */
    public void traverseDir(Vector fileVect, WildCardFilter filter, String folderName, boolean
useSubFolders) {
        if ( folderName != null ) {
            File folder = new File(folderName);
            File[] fileList = folder.listFiles();
            if ( fileList != null ) {
                for ( int i = 0; i < fileList.length; i++ ) {
                    if ( !fileList[i].isDirectory() ) {
                        if ( filter.accept(folder, fileList[i].getName()) ) {
                            try {
                                fileVect.add(fileList[i].getCanonicalPath());
                            }
                            catch(Exception e) {
                                fileVect.add(fileList[i].getAbsolutePath());
                            }
                        }
                    }
                    else if ( useSubFolders ) {
                        try {
                            traverseDir(fileVect, filter, fileList[i].getCanonicalPath(),
useSubFolders);
                        }
                        catch(Exception e) {
```

```
                            traverseDir(fileVect, filter, fileList[i].getAbsolutePath(),
useSubFolders);
                        }
                    }
                }
            }
        }
    }

    /**
     * Method for reading a file from disk.<p>
     * @param absoluteFileName name of the file (inc. its absolute path name)
     * @param modTime the value used to check against the file modification time to determine
whether or not to pickup
     * @param doreg controls whether it was called from the collect method internally or from
a RMI call
     * @return FSFromReadFile information passed back from this method
     * @exception AdapterException
     * @since Woodstock 2.0
     */
   public FSFromReadFile readFile(String absoluteFileName, int modTime, boolean doreg) throws
AdapterException {
        if ( doreg ) {
            registerThread();
        }
        FSFromReadFile frf = new FSFromReadFile();
        if ( LogService.out.debug ) {
            LogService.out.logDebug("Trying to read file " + absoluteFileName);
        }
        File file = new File(absoluteFileName);
        if ( !file.canRead() ) {
            if ( LogService.out.debug ) {
                LogService.out.logDebug("can't read file");
            }
            if ( doreg ) {
                unregisterThread();
            }
            frf.data = CANTREAD;
            return frf;
        }
        final long lastmod = file.lastModified();
        final long curtime = System.currentTimeMillis();
        if ( curtime - lastmod < modTime ) {
            if ( LogService.out.debug ) {
                LogService.out.logDebug("Modification time still under limit");
            }
            if ( doreg ) {
                unregisterThread();
            }
            return frf;
        }
        final int fileSize = (int)file.length();
        int bytesRead = 0;
        int bufSize = BUFFER_SIZE;
        ByteArrayOutputStream baoStream = new ByteArrayOutputStream(fileSize);
        BufferedInputStream biStream = null;

        if ( fileSize > 0 ) {
            try {
                if ( fileSize < bufSize ) {
                    bufSize = fileSize;
                }
```

```
                biStream = new BufferedInputStream(new FileInputStream(file), bufSize*2);
                frf.data = new byte[bufSize];

                while ( (bytesRead = biStream.read(frf.data)) != -1 ) {
                    baoStream.write(frf.data, 0, bytesRead);
                }
                frf.data = baoStream.toByteArray();
            }
            catch(Exception e) {
                if ( doreg ) {
                    unregisterThread();
                }
                LogService.out.logException("Exception in readFile:", e);
                throw new AdapterException(e);
            }
            finally {
                if ( biStream != null ) {
                    try {
                        biStream.close();
                    }
                    catch(Exception e) {
                        LogService.out.logWarn(e.getMessage());
                    }
                }
            }
        }

        // check if file has been modified since read
        if ( file.lastModified() == lastmod ) {
            frf.goodfile = true;
        }
        if ( doreg ) {
            unregisterThread();
        }
        return frf;
    }

    /**
     * Method for writing a file to disk.<p>
     * @param tei a FSToExtractInfo class containing all the necessary parms
     * @exception AdapterException
     * @since Woodstock 2.0
     */
    public void writeFile(FSToExtractInfo tei) throws AdapterException {
        registerThread();
        BufferedOutputStream boStream = null;
        try {
            String filename = null;
            String folder = new File(tei.folder).getCanonicalPath();
            if ( tei.assignedFilename != null && tei.assignedFilename.length() > 0 ) {
                tei.assignedFilename = new File(tei.assignedFilename).getName();
                filename = folder + java.io.File.separator +
Util.getUniqueFileName(tei.assignedFilename);
            }
            else {
                filename = folder + java.io.File.separator + new File(tei.bodyName).getName();
            }
            boStream = new BufferedOutputStream(new FileOutputStream(filename, tei.append));
            if ( tei.data != null ) {
                boStream.write(tei.data, 0, tei.data.length);
                boStream.flush();
            }
```

```
            boStream.close();

            // Workaround for websphere bug
            String rfp = Manager.getProperty("resetFilePerms");
            if ( "true".equalsIgnoreCase(rfp) || "yes".equalsIgnoreCase(rfp) ) {
                String mask = Manager.getProperty("resetFilePermsMask");
                if ( mask == null ) {
                    mask = "664";
                }
                String os = System.getProperty("os.name");
                if ( os != null && os.toLowerCase().indexOf("windows") == -1 ) {
                    Runtime.getRuntime().exec(new String[]{"chmod", mask, filename});
                }
            }
        }
        catch(Exception e) {
            unregisterThread();
            throw new AdapterException(e);
        }
        finally {
            if ( boStream != null ) {
                try {
                    boStream.flush();
                    boStream.close();
                }
                catch(Exception e) {
                    LogService.out.logWarn(e.getMessage());
                }
            }
        }
        unregisterThread();
    }

    /**
     * Method for deleting a file on disk
     * @param absoluteFileName name of the file with its absolute path
     * @exception AdapterException
     * @since Woodstock 2.0
     */
     public void deleteFile(String absoluteFileName) throws AdapterException {
        registerThread();
        File file = null;
        try {
            file = new File(absoluteFileName);
            file.delete();
        }
        catch(Exception e) {
            unregisterThread();
            throw new AdapterException(e);
        }
        unregisterThread();
     }

    /**
     * Method used to import a file directly
     * @param dirname the directory name
     * @param filename the filename
     * @return FSFromReadFile information passed back from this method
     * @since Woodstock 2.2
     */
    public FSFromReadFile importFile(String dirname, String filename) throws AdapterException
{
```

```
    registerThread();
    String fileToRead = filename;
    if ( dirname != null && dirname.length() > 0 ) {
        try {
            fileToRead = new File(dirname, filename).getCanonicalPath();
        }
        catch(Exception e) {
            unregisterThread();
            throw new AdapterException(e);
        }
    }

    FSFromReadFile frf = null;
    try {
        frf = readFile(fileToRead, DEF_MODTIME, false);
    }
    catch(Exception e) {
        unregisterThread();
        throw new AdapterException(e);
    }
    frf.importFile = fileToRead;
    unregisterThread();
    return frf;
}

/**
 * Method used to export a file directly
 * @param filename the filename
 * @since Woodstock 2.2
 */
public void exportFile(String filename, byte[] data) throws AdapterException {
    registerThread();
    BufferedOutputStream boStream = null;
    try {
        boStream = new BufferedOutputStream(new FileOutputStream(filename));
        if ( data != null ) {
            boStream.write(data, 0, data.length);
            boStream.flush();
        }
        boStream.close();

        // Workaround for websphere bug
        String rfp = Manager.getProperty("resetFilePerms");
        if ( "true".equalsIgnoreCase(rfp) || "yes".equalsIgnoreCase(rfp) ) {
            String mask = Manager.getProperty("resetFilePermsMask");
            if ( mask == null ) {
                mask = "664";
            }
            String os = System.getProperty("os.name");
            if ( os != null && os.toLowerCase().indexOf("windows") == -1 ) {
                Runtime.getRuntime().exec(new String[]{"chmod", mask, filename});
            }
        }
    }
    catch(Exception e) {
        unregisterThread();
        throw new AdapterException(e);
    }
    finally {
        if ( boStream != null ) {
            try {
                boStream.flush();
```

```
                    boStream.close();
                }
                catch(Exception e) {
                    LogService.out.logWarn(e.getMessage());
                }
            }
        }
        unregisterThread();
    }
}
```

FSFromCollectInfo File

The following sample code is an FSFromCollectInfo file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.util.Vector;

/**
 * Information passed from the collect method
 * @since Woodstock 2.1
 */
public class FSFromCollectInfo implements java.io.Serializable {
    public int threadCnt = 0;
    public int filesCollected = 0;
    public boolean couldntRead = false;
    public Vector workflows = null;

    public FSFromCollectInfo() {}

    public synchronized void addBootstrapWF(String wfId) {
        workflows.add(wfId);
    }

    public synchronized void fileCounter() {
        filesCollected++;
    }

    public synchronized void threadCounter(boolean add) {
        if ( add ) {
            threadCnt++;
        }
        else {
            threadCnt--;
```

FSFromReadFile File

The following sample code is an FSFromReadFile:

```
package com.sterlingcommerce.woodstock.services.filesystem;

/**
 * Information passed from the readFile method
 * @since Woodstock 2.1
 */
public class FSFromReadFile implements java.io.Serializable
{
    byte[] data = null;
    boolean goodfile = false;
    String importFile = null;

    public FSFromReadFile() {}
}
```

FSToCollectInfo File

The following sample code is an FSToCollectInfo file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.util.Vector;
import com.sterlingcommerce.woodstock.workflow.WFCTransportMessage;

/**
 * Information passed to the collect method
 * @since Woodstock 2.1
 */
public class FSToCollectInfo implements java.io.Serializable {
    public int iwfid = -1;
    public int modTime = 0;
    public int maxThreads = 10;
    public boolean useSubFolders = false;
    public boolean keepPath = false;
    public boolean delete = true;
    public boolean getzero = false;
    public String folder = null;
    public String filter = null;
    public String svcName = null;
    public WFCTransportMessage wfctm = null;
    public FSToCollectInfo() {}
}
```

FSToExtractInfo File

The following sample code is an FSToExtractInfo file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.util.Vector;

/**
 * Information passed to the collect method
 * @since Woodstock 2.1
 */
public class FSToExtractInfo implements java.io.Serializable
{
    public boolean append = false;
    public String folder = null;
    public String bodyName = null;
    public String assignedFilename = null;
    public byte[] data = null;

    public FSToExtractInfo() {}
}
```

WFStartThread File

The following sample code is a WFStartThread file:

```
package com.sterlingcommerce.woodstock.services.filesystem;

import java.io.File;
import java.util.Arrays;
import com.sterlingcommerce.woodstock.services.XLogger;
import com.sterlingcommerce.woodstock.workflow.InitialWorkFlowContext;
import com.sterlingcommerce.woodstock.workflow.WorkFlowContextCookie;

class WFStartThread implements Runnable {
    private String filename = null;
    private FSToCollectInfo tci = null;
    private FSFromCollectInfo fci = null;
    private FileSystemServerImpl parentThread = null;

    public WFStartThread(String filename, FSToCollectInfo tci, FSFromCollectInfo fci,
FileSystemServerImpl fsImpl) {
        this.filename = filename;
        this.tci = tci;
        this.fci = fci;
        parentThread = fsImpl;
    }

    public void run() {
        String msg = "Exception reading file"; // initialize to first possible exception
        try {
            FSFromReadFile frf = parentThread.readFile(filename, tci.modTime, false);
            if ( frf.goodfile ) {
```

```
                if ( frf.data == null && tci.getzero ) {
                    frf.data = "".getBytes(); // allows for collecting zero byte files
                }
                // don't do an 'else' here because frf.data might have just been set above
                if ( frf.data != null ) { // only collect if data not null at this point
                    String file2del = filename; // in case path is stripped off
                    if ( !tci.keepPath ) {
                        filename = new File(filename).getName();
                    }
                    InitialWorkFlowContext iwfc = new InitialWorkFlowContext();
                    iwfc.addContentElement("FileName", filename);
                    iwfc.setWorkFlowDefId(tci.iwfid);
                    iwfc.setDocumentName(filename);
                    iwfc.setDocumentBody(frf.data);
                    iwfc.setMessageToChild(tci.wfctm);
                    msg = "Exception starting workflow";
                    WorkFlowContextCookie wfcCookie = iwfc.start();
                    if ( wfcCookie != null ) {
                        fci.addBootstrapWF(Long.toString(wfcCookie.getWorkFlowId()));
                    }
                    fci.fileCounter(); // synchronized increment

                    if ( tci.delete ) {
                        msg = "Exception deleting file";
                        File file = new File(file2del);
                        file.delete();
                    }
                }
            }
            else if ( frf.data != null && Arrays.equals(frf.data,
FileSystemServerImpl.CANTREAD) ) {
                fci.couldntRead = true;
            }
        }
        catch(Exception e) {
            XLogger log = new XLogger("WFStartThread", tci.svcName);
            log.logException(msg, e);
        }
        finally {
            fci.threadCounter(false); // synchronized decrement
```

Index

I

J

L

M

O

P

R

S

T

U

W

X