Using IBM WebSphere Service Registry and Repository Feature Pack for Service Federation Management to Share Services from an SAP Domain

This IBM® Redpapers™ publication provides a detailed explanation of how IBM WebSphere® Service Registry and Repository Feature Pack for Service Federation Management (SFM) can be used to share services from a non-IBM application server, in this case SAP. It focuses primarily on the interactions required with SAP to achieve this functionality.

This paper assumes prior knowledge of SFM, and so does not provide details about SFM components and terminology, or detailed step by step instructions for interacting with SFM. However, if you are already familiar with SFM, this paper should provide all of the information required for you to create a new SFM domain that contains services hosted in SAP, add the domain to an existing SFM federation, and share services from the new domain to another domain.

*Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 provides an introduction to SFM, and also provides detailed step by step instructions for all of the interactions required in this paper. If you are not familiar with SFM, *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 should be considered required reading.

This paper is based on a real-world scenario, following a fictitious company with a business requirement to share SAP services.
What is the goal of Service Federation Management

Service-oriented architecture (SOA) has been widely adopted as the underpinning of enterprise architectures. Much of the value of SOA derives from service reuse and the flexibility and agility offered by a set of reusable services. Historically, service reuse has started within a particular business unit, and this business unit is often referred to as a service domain.

Many enterprises are now moving beyond SOA projects that focus on specific, departmental business problems within a single service domain, to more complex SOA installations that extend the reach of SOA, such that it supports end-to-end business interactions across business unit boundaries. These enterprises are effectively scaling up their success with SOA. From a business perspective, they are scaling up or extending service reuse, so that existing and new services can be reused across service domains in the enterprise.

Service Federation Management simplifies the process of reusing business services across multiple and heterogeneous service domains. The SFM Console provides a web browser based graphical user interface, running in Business Space, which allows the user to take a business view of their enterprise. They are able to specify the services they want to share, and the domains they want to share to, without being swamped by technical details about the services or domains.

SFM uses the Service Connectivity Management Protocol, which is implemented in the WebSphere Service Registry and Repository Feature Pack for Service Federation Management, WebSphere Enterprise Service Bus, and WebSphere Message Broker. This implementation allows SFM to drive the copying of service metadata and the creation of service proxies in the underlying products. The business value of SFM lies in facilitating these services while simplifying the complexity associated with doing this task manually.

SAP components and terminology

This section describes the core terms and components used in this paper.

Standards based terms and technologies

Here are the terms and technologies used in this paper:

- Extensible Markup Language (XML): A generic syntax that facilitates the creation of structured text that strongly describes a data structure in a format that is both human and machine readable.
- Web Service Definition Language (WSDL): An XML based syntax that is used to record and describe all aspects about a given web service. This includes a description of its function and its input and output message format.
- Simple Object Access Protocol (SOAP): An XML based syntax that is used to carry the input or output payload of a web service over a wire protocol, such as HTTP.
- Service-oriented architecture (SOA): A design philosophy to structure an IT deployment around discrete components than can later be easily reused and woven into new functions.
- Universal Description Discovery and Integration (UDDI): An XML based web service repository that stores the WSDL metadata in a structured and searchable format that assists in life cycle management.
Java™ Platform, Enterprise Edition (Java EE / J2EE): A Java based set of technologies focused on tooling Java for server-based computing environments.

Enterprise Java Bean (EJB): A Java based technology for designing container managed components to aid in composition of modular applications. It aids in the encapsulation of business logic code into easily digested components.

Enterprise Archive (EAR): A compressed archive used to package Java based applications for deployment on to a Java EE based server.

SAP components

The following SAP components are described in this paper.

- **SAP NetWeaver**: An Enterprise Resource Planning (ERP) system that has been migrating into the world of SOA. It consists of a mesh of databases and application servers hosting business applications.

- **SAP NetWeaver Application Server ABAP**: Allows for the composition and execution of programs written in an SAP developed language similar to COBOL called ABAP. ABAP services can be exposed as web services to facilitate a transition to SOA.

- **SAP NetWeaver Application Server Java**: Consists of an HTTP web server and a Java EE Application Server.

- **SAP NetWeaver UDDI Registry**: SAP Application Server Java hosts a Universal Description Discovery and Integration (UDDI) Server and Client that are compliant with the OASIS UDDI V3.0 specification.

- **SAP NetWeaver Services Registry**: SAP Application Server Java hosts an extension of the UDDI standard to allow for a more detailed categorization and governance of web services. It works by adding additional metadata and life cycle management facilitates on top of the existing UDDI Registry. Use this mechanism for management of web services as opposed to the UDDI Registry Client directly.

- **SAP NetWeaver Developer Studio**: An Eclipse-based Integrated Development Environment (IDE) that allows for the development, testing, and deployment of Java-based applications into SAP Application Server Java.

- **SAP Developer Network (SDN)**: Hosted directly by SAP AG server, SAP Developer Network provides a community of wikis, tools, resources, product manuals, and downloads for SAP users and developers. The SDN also serves as the single portal to locate and download SAP Notes, Installation Files, Patches, and various other SAP related files.

Introduction to SAP

SAP AG is a multinational software company based in Waldorf, Germany. SAP is known for Enterprise Resource Planning (ERP). SAP was originally a client-server application powered by a COBOL based language called ABAP. However, SAP recently adopted Java as the language of choice for their new releases.

An SAP installation is characterized by a core SAP NetWeaver known as the *SAP kernel*. The kernel is extended by one or more SAP Application Server ABAP or SAP Application Server Java. SAP Application Server Java is a Java EE based application server. For Java development, SAP NetWeaver Development Studio is used.
The scenario used in this paper

ITSO T-Shirt Store is a fictitious company that is used in this paper to provide a real-world business scenario where we can use SFM's capabilities. ITSO T-Shirt Store is a t-shirt retailer, which has both a physical and online store.

ITSO T-Shirt Store has just purchased a company that sells socks, and rebranded it as the ITSO Sock Store. ITSO T-Shirt Store wishes to reuse some of the services of ITSO Sock Store to save time and effort on developing new services. ITSO Sock Store currently uses SAP Application Server to host its services, and SAP NetWeaver UDDI Registry, from now on referred to as SAP UDDI, to store its WSDL documents.

ITSO T-Shirt Store is already using SFM to share services between domains. In particular, it has a Retail domain, and an eBusiness domain with services shared between them. All of its domains use IBM WebSphere Application Server to host the web services. ITSO T-Shirt Store uses WebSphere Application Server administrative security across all of its domains.

Note: The introduction of SFM to ITSO T-Shirt Store is covered in Strengthening Your ESB with WebSphere Service Registry and Repository, REDP-4686.

Within ITSO Sock Store, there are a number of existing service domains, including one known as the Sock domain. The Sock domain contains the SAP servers running the services to be shared. In this paper, we create an SFM domain called Sock domain for this service domain. The additional software that is required to make the services available to SFM will be installed and configured in this domain. SFM is then be used to shared these services to ITSO T-Shirt Store’s existing eBusiness domain.

Within the eBusiness domain, ITSO T-Shirt Store uses WebSphere Message Broker as the connectivity provider. In this paper, we use WebSphere Enterprise Service Bus as the connectivity provider in the Sock domain.

Note: Currently, as connectivity providers within SFM, WebSphere Enterprise Service Bus and WebSphere Message Broker offer the same level of functionality. Either product could be used as the connectivity provider in the Sock domain.
The final infrastructure

Figure 1 shows how the systems will look after the steps in this paper have been completed.

The original web services are hosted in the SAP Application Server in the Sock domain. The WSDL files representing these web services are hosted in the SAP UDDI. WebSphere Service Registry and Repository, labelled WSRR, is installed in the Sock domain, and is configured as a registry server and a domain server. The WSDL files for the services are also in the registry server. WebSphere Enterprise Service Bus is installed and configured as the connectivity provider for this domain, and contains an SFM proxy.

In the eBusiness domain, WebSphere Message Broker, labelled WMB, is the connectivity provider, and has an SFM proxy. WebSphere Service Registry and Repository is configured as the registry server and domain server for the eBusiness Domain. The WSDL documents representing the shared web services are in the registry server. This WebSphere Service Registry and Repository is also being used for the federation server and the SFM Console.

Note: In this paper, the federation server and the SFM Console are configured on WebSphere Service Registry and Repository in the eBusiness domain. Note that in Strengthening Your ESB with WebSphere Service Registry and Repository, REDP-4686 that they were in the Retail domain.
Figure 1 on page 5 shows the final infrastructure in terms of the software involved. Figure 2 focuses on just one of the web services, named A for convenience, and shows how the web service is shared by SFM, and subsequently, how a service consumer in the eBusiness domain could use web service A.

The original web service is A, in the blue box in the Sock domain. (This service is hosted in SAP Application Server). The orange document labelled A in the SAP UDDI is the WSDL document representing this web service. The same orange document representing the WSDL document also exists in the registry server. A proxy, labelled A’, is created in the WebSphere Enterprise Service Bus, labelled WESB. This proxy calls the original web service A, therefore providing a new endpoint for the original service A. The orange document labelled A’ in the registry server in the Sock domain is the WSDL for this proxy. The same WSDL, representing the proxy A’, is also in the registry server in the eBusiness domain. The eBusiness domain also has a proxy, labelled A”, which in this case is hosted in WebSphere Message Broker, labelled WMB. This proxy calls the proxy in the Sock domain. The WSDL for this proxy is also stored in the registry server. The blue box on the right side in the eBusiness domain represents a service consumer that wants to use web service A.

After SFM has shared the service and created the layout shown in Figure 2, a service consumer in the eBusiness domain can use the original service A that is in the Sock domain as follows:

1. The service consumer looks up the service they want in the WebSphere Service Registry and Repository in the eBusiness domain.
2. They find the endpoint for A”.
3. They make a call to the endpoint for A”. This is a call to the proxy in the eBusiness domain.
4. The eBusiness domain proxy makes a web services call to the proxy A’ in the Sock domain.
5. The Sock domain proxy makes a call to the original web service A hosted in SAP Application Server.
6. The response is passed back through the same chain to the service consumer in the eBusiness domain.
Preparing Service Federation Management for use

This section describes the software installation and configuration required for the SFM environment created in this paper.

Software versions

This paper uses the software versions shown in Table 1.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Service Registry and Repository</td>
<td>7.0.0.2</td>
</tr>
<tr>
<td>SFM Feature Pack</td>
<td>7.0.0.1</td>
</tr>
<tr>
<td>WebSphere Enterprise Service Bus</td>
<td>7.0.0.2</td>
</tr>
<tr>
<td>WebSphere MQ</td>
<td>7.0.1</td>
</tr>
<tr>
<td>WebSphere Message Broker</td>
<td>7.0.</td>
</tr>
<tr>
<td>SAP NetWeaver</td>
<td>7.1</td>
</tr>
<tr>
<td>SAP Application Server Java</td>
<td>7.1</td>
</tr>
<tr>
<td>SAP NetWeaver Development Studio</td>
<td>7.1</td>
</tr>
</tbody>
</table>

This paper uses Windows® XP Service Pack 3 as the operating system.
Prerequisite infrastructure

This paper assumes that you have an initial infrastructure that matches the current infrastructure of ITSO T-Shirt Store. This infrastructure is shown in Figure 3. This section provides details about how to configure this infrastructure.

Within the Sock domain, ITSO T-Shirt Store has an SAP Application Server hosting the two web services that we want to share. The WSDL documents for these services are held in the SAP UDDI. In the eBusiness domain, SFM is already installed, configured, and being used. A WebSphere Service Registry and Repository is being used as the registry server, domain server, federation server, and for the SFM Console. WebSphere Message Broker is installed and configured as the connectivity provider in this domain.

Note: In the scenario in this paper, ITSO T-Shirt Store already uses basic WebSphere administrative security, and so it is used in the Sock domain also. It is possible to use SFM without security.

Note: Several machines will likely be required to complete this scenario. In a production environment, it may be the case that each product would actually be on its own machine, with the machines hosted in their particular domains. However, in this paper, the software products can be installed however you want, for example, with one machine for each domain, or on multiple machines. Use at least two machines, one for each domain. It is important that you remember which product you consider to be in each domain.
Setting up the Sock service domain
Perform the following steps to install and configure the SAP products, and install the web services that will be shared.

ITSO Sock Store uses an SAP based infrastructure to manage its inventory and order procurement systems. In order to mimic its setup, you need to have access to an SAP system with all the requisite software and user accounts. You need to have a kernel version of SAP NetWeaver 7.1 running at least one SAP Application Server Java with both of the SAP NetWeaver UDDI Registry and SAP NetWeaver Services Registry modules embedded.

For the purposes of this paper, you do not need to be an SAP expert. You need to work with your company’s SAP administrator. With the administrator’s help, you should be able to complete all of the SAP interactions needed for this paper. From an SAP perspective, this paper focuses on the steps needed to deploy and consume Java-based web services on SAP through the UDDI Registry.

Administrator tasks
Work with your SAP administrator to perform the following steps:

1. Set up an SAP System with at least the following software:
   a. SAP NetWeaver 7.1.
   b. SAP Application Server Java.
   c. SAP NetWeaver UDDI Registry.
   d. SAP Cryptographic Library. (Optional, but you need it if you want to use SSL encryption.)

2. Set up a user account to the SAP Developer Network (SDN). You can access the SDN by going to the following address:
   http://www.sdn.sap.com

3. Set up a user account to the SAP Application Server Java:
   a. The account must have Java EE administrator rights.
   b. The account is usually J2EE_ADMIN.
   c. The password is assigned by your SAP administrator.

4. Have your SAP administrator provide you with the host name and TCP/IP port on which your SAP Application Server Java is running. These two items are used to build the base URL needed to access the SAP NetWeaver administration web pages. The general format of the URL is:
   http://<sap_host>:<port>/<url_path>
   Here is an example URL:

5. Review, confirm, and apply the fix recommended in the SAP Note, 1258505. This SAP Note describes an issue in the SAP UDDI implementation that causes some issues with WebSphere Service Registry and Repository. The issue either requires the installation of a patch or that your system be running the latest release of the SAP UDDI Server. Without the patch, there could be some duplicate entries and incorrect user associations.

6. Provide access to the installation archive for the Integrated Development Environment (IDE) called SAP NetWeaver Development Studio 7.1. You need to use your SDN account and directly download SAP NetWeaver Development Studio 7.1 by going to the following address:
   https://nwds.sap.com/swdc/downloads/updates/netweaver/nwds/ce/
Installing SAP NetWeaver Development Studio 7.1

You need to use a Java based Integrated Development Environment (IDE) called Eclipse. Eclipse is an open source project that provides a clean and robust framework into which advanced application specific features can be built. In this case, SAP has taken the base Eclipse environment and added its own framework, allowing them to re-brand the tool as SAP NetWeaver Development Studio (NWDS). We use SAP NetWeaver Development Studio 7.1.

To install NWDS, perform the following steps:

1. Obtain the SAP NetWeaver Development Studio 7.1 installation archive named NWDS_platform.zip (the archive should be about 101 MB in size).
2. Extract the archive into a temporary directory.
3. SAP NetWeaver Development Studio does not support IBM based Java and the installer fail if it is directed to use one. Ensure that your workstation has Sun Java Development Kit (JDK) 1.6 installed. Then configure your environmental variables %JAVA_HOME% and %PATH% to the path of the JRE root and JDK bin directories, respectively.
4. Start the SAP NetWeaver Development Studio installation wizard by running the application called IDESetup.exe.
5. There will be a prompt asking for the URL to the update site. Use the default URL:
   https://nwds.sap.com/swdc/downloads/updates/netweaver/nwds/ce/
6. You are prompted to enter the file path to your Sun Java based Java Development Kit (JDK) 1.6 installation. Use the default file path.
7. At the end of the first part of the installation, you see the first run of the SAP NetWeaver Development Studio platform. The installation is not yet complete at this point. A second installation part runs where the Eclipse based update manager runs. The Eclipse Update Manager is used to upgrade and install features into Eclipse. In this case, SAP uses the tool as a means to upgrade their static installer with the latest builds from their servers.
8. There will be a prompt requesting your SDN account details as the Update Manager runs. Follow the Update Manager steps and select all the features for installation. The cumulative download size for these features should be somewhere around 600 MB.
9. After the Update Manager has completed, you are prompted to restart Eclipse or apply the changes now. Choose the option to restart Eclipse.
10. You should have SAP NetWeaver Development Studio 7.1 installed on your workstation. When Eclipse has restarted, the installation should look like Figure 4.

![NetWeaver Development Studio welcome page](image)

**Figure 4  NetWeaver Development Studio welcome page**

**Adding SAP Application Server into SAP NetWeaver Development Studio**

Your installation of SAP NetWeaver Development Studio is fresh and has no knowledge of your specific SAP system and Application Server Java. Apply the following settings to inform SAP NetWeaver Development Studio about the location of the SAP system.

1. Open SAP NetWeaver Development Studio 7.1.
2. Select Window → Preferences → SAP AS Java.
3. Under SAP Instance Identity, enter values for Instance host (the SAP host name) and Instance number (the system number).
4. Click Register SAP Instance.
5. After a short while, you see your SAP system show up in the SAP System Instances list. It should also be automatically selected as your default SAP system (Figure 5).

![Figure 5 Preferences window]

6. Click OK.

7. Open the SAP Management Console perspective. Select **Window → Open Perspective → Other** and, in the Open Perspective window, select **SAP Management Console**. Click **OK**. This perspective shows an overview of your SAP System and its myriad of functions and active processes.

**Importing the ITSO Sock Company workspace**

Your workspace is currently empty and has nothing to deploy into SAP. You must import an existing workspace that has a mock web service for the ITSO Sock Company. This workspace is provided in the additional materials provided with this paper (see “Additional material” on page 46) in a file named `ITSO_Sock_Company Workspace.zip`. This workspace consists of an EJB project called `ITSO_Sock_Company` and its wrapping EAR. These projects are fully implemented and only require deployment into SAP to become functional.

Perform the following steps:

1. Select **File → Import**.
2. Expand **General** and select **Existing Projects into Workspace** (Figure 6).

![Figure 6 Import existing workspace](image)

3. Click **Next**.
4. Select the **Select archive file** option.
5. Click **Browse** and select **ITSO Sock Company Workspace.zip**.
6. Confirm that both the ITSO_Sock_Company and ITSO_Sock_Company_EAR projects are selected (Figure 7).

![Figure 7 Selecting import](image)

7. Click **Finish**. Wait a few seconds for the project to be imported and built.
8. Select **Window → Open Perspective → Java EE** to open the Java EE perspective.
9. You should now see the ITSO Sock Company projects in your workspace. Expand **ITSO_Sock_Company -> ejbModule -> com.itso.sock** and double-click **Sock.java** to open it in the Java editor (Figure 8). This is the simple Java class used to implement the Sock web service.

![Image of workspace extending from page](image)

**Figure 8** The new workspace

**Deploying the EAR into SAP Application Server Java**

Your workspace now has a functional project with an implemented web service called Sock. Deploy this Sock web service into SAP Application Server Java by performing the following steps:

1. Verify that Sap NetWeaver Development Studio is connected to SAP AS Java. Select the **Servers** view. Ensure the SAP Server `<system_id>` is in a Started state.

2. If the status is Republish, right-click the server and select **Republish**. The status should now show Synchronized.

3. Proceed to deploy the EAR into SAP AS. Right-click the server and select **Add and Remove Projects**.

4. Under Available Projects, select **ITSO_Sock_Company_EAR** and click **Add**.
5. The ITSO_Sock_Company_EAR should now appear in the Configured Projects field (Figure 9).

![Add and Remove Projects window](image)

**Figure 9** Add and Remove Projects window

6. Click **Finish**.

7. As the EAR is deployed, you are prompted to enter the login details for your SAP System. Enter your J2EE_ADMIN credentials (Figure 10).

![Entering credentials](image)

**Figure 10** Entering credentials
8. The EAR is now in the process of being deployed. After a moment, you see a window that informs you that the deployment finished successfully (Figure 11).

![Deployment successful message](image)

**Figure 11** Deployment successful message

9. Click **OK**.

10. Now verify that the EAR was successfully deployed. In the Servers view, select your server SAP Server `<system_id>` and expand all of its children. You should see a Synchronized status for SAP, ITSO_Sock_Company_EAR, and ITSO_Sock_Company (Figure 12).

![Verifying that the EAR is deployed](image)

**Figure 12** Verifying that the EAR is deployed

11. You have now deployed a Java-based web service into SAP AS Java.

**Accessing the web portal for SAP Application Server Java**

SAP has built a series of web-based portals to aid in the management of their SAP Application Server Java and it associated modules. For the next few sections, we use these portals to review and manage the settings of the Sock web service.

Open a web browser and navigate to:

http://<hostname>:50000/index.html
This page is an index of various operations that you can perform against the sub-systems of the SAP Application Server Java. If you click any of the icons, you are taken to the portal for that application.

**Viewing and testing a web service on SAP AS Java**
The Sock web service is now deployed and running on SAP Application Server Java. You should now review the Web Services Navigator to verify and test the Sock web service.

Perform the following steps:
1. Open a web browser and navigate to:
   http://<sap_host>:50000/index.html
2. Select the **Web Services Navigator** icon.
3. You are prompted to log in (Figure 14). You should use your J2EE_ADMIN user ID and enter your password, and then click **Log on**.

![Login pane](image1.png)

**Figure 14**  Login pane

4. The page should be titled SAP NetWeaver WS Navigator and is on the Select Service tab. Click **Search by Logical Destination**.

5. In the **Search for** field, enter Sock and click **Search**. Wait a moment while the search is performed. You should now see a result named Sock (Figure 15). Select it.

![Sock operations](image2.png)

**Figure 15**  List Sock operations
6. A new page should open in the Select Operation tab. Notice the URL pointing to the WSDL for the Sock web service. Make a note of the URL to the WSDL for later. Click the WSDL URL.

7. A new window should open that shows the WSDL's XML. Take a quick look at this XML and then go back to the previous window.

8. Click the checkStock operation.

9. A new page should open in the Enter Input Parameters tab. This page allows you to invoke the Sock web service using your own input values. Enter a positive integer value as the order number. For example, you could use the order number 1234 (Figure 16).

![SAP NetWeaver WS Navigator](image)

**Figure 16** Inputting a parameter
10. Click **Execute**. The Result tab opens. This tab shows you exactly what actions were performed and the result returned. It also shows the raw SOAP messages used for both request and result (Figure 17). The returned result displays the amount of socks for type 1234 that the store has in stock.

![Figure 17 Result of the Sock test](image)

You have now verified and tested a web service on SAP Application Server Java.

**Registering and validating the web service into the SAP Service Repository**

In accordance with SAP guidelines, we follow the recommended route and publish the Sock web service WSDL using the SAP Service Repository. This action also automatically publishes the web service into the UDDI Registry. The SAP Service Repository adds additional metadata on top of the UDDI Registry.

Perform the following steps:

1. Open a web browser and navigate to:
   
   *http://<sap_host>:50000/index.html*

2. Select the **Services Registry** icon and log in.

3. The resultant page should be titled Services Registry. Select the **Publish** tab.
4. In the Endpoint WSDL field, enter the URL to the Sock WSDL (Figure 18).

![Service Registry](image)

Figure 18  Publish WSDL

5. Click **Publish**. Wait a moment while the web service is added to the Service Registry. You have published the WSDL into the Service Registry.

6. Proceed to verify that the WSDL was added successfully. Click the **Service Definitions** tab. In the Find field, enter Sock and click **Go**. Wait a moment as SAP searches its registry.
7. You should now see a single entry under WSDL Port Type Name called Sock. Select it. You should now see a detailed view that describes the metadata stored for the Sock web service (Figure 19).

![Service Registry](image)

Figure 19  Verify WSDL

You have now published a WSDL into the SAP Service Repository and then verified that it was added successfully.

**Validating the web service in SAP NetWeaver UDDI Registry**

Now verify that the SAP UDDI Registry was updated by the SAP Service Repository and is now hosting the reference to the Sock web service.

Perform the following steps:

1. Open a web browser and navigate to:
   
   ```plaintext
   http://<sap_host>:50000/index.html
   ```
2. Select the **UDDI Client** icon and log in.
3. The page that opens should be titled SAP NetWeaver UDDI Client v3 Registry. In the Name field, enter Sock (Figure 20) and click **Find**.

![Finding the Sock web service](image)

**Figure 20** Finding the Sock web service

4. Wait as SAP queries the UDDI tModel for a service called Sock. You should see the page refresh and a single entry for a tModel matching Sock. You see that the entry has a UDDI Key and a name (Figure 21). Select the UDDI Key.

![tModel Details](image)

**Figure 21** tModel Details
You should be able to view the details about the UDDI tModel for the Sock web service (Figure 22)

![Figure 22 Verifying the web service](image)

Gathering the UDDI API URLs from SAP NetWeaver UDDI Registry

To extract the UDDI metadata from SAP, you need to gather the links to the UDDI API URLs. These API URLs are used by WebSphere Service Registry and Repository to retrieve the SAP services.

Perform the following steps:

1. In the UDDI Client, click Settings.
2. A window opens and prompts you to log in again. Use the J2EE_ADMIN User ID and enter your password (Figure 23), and click Log In.

![Figure 23 Login window](image)
3. The Connection Settings page opens. This page lists the various API URLs for the SAP built-in UDDI Serve (Figure 24). These API URLs can allow other UDDI enabled clients to consume services from this service registry.

![API URLs](image.jpg)

**Figure 24**  API URLS

4. Take note and write down all of the API URLs:
   - Inquiry: http://localhost:50000/uddi/api/inquiry
   - Publish: http://localhost:50000/uddi/api/publish
   - Subscription: http://localhost:50000/uddi/api/subscription

5. Notice that the URLs are using the localhost host name. You must modify the URLs so that they are fully qualified with the SAP host's domain name. In our example, they URLs should be:
   - Publish: http://sapnw71.svl.ibm.com:50000/uddi/api/publish
   - Subscription: http://sapnw71.svl.ibm.com:50000/uddi/api/subscription

You have now gathered the API URLs from the SAP UDDI v3 Registry.

**Gathering the UDDI API access policy details**

You now have the APIs where you direct UDDI requests to, but not the specific details about security and access protections that they might have. These policy, security, and token details must match exactly when you attempt to import the SAP services into IBM.

Perform the following steps:

1. Open a web browser and navigate to:
   
   http://<sap_host>:50000/index.html

2. Select **SAP NetWeaver Administrator** icon and log in.

3. The page that opens is titled SAP NetWeaver Administrator. Click **SOA Management** in the top menu and click **Technical Configuration** in the sub-menu.
4. Click the **Service Registry Configuration** icon in the main body (Figure 25).

5. The page that opens is titled Services Registry Configuration: UDDI Server Configuration. Click the **Policy** tab.

6. The Policy tab shows all the details and security restrictions placed on the UDDI APIs (Figure 26). You need to reference back to this page later in this paper. Make a screen capture or printout of this page for future reference.

**Summary**

At the start of this paper, the assumption was that you had an clean SAP system with no deployed web services. However, we assume that you are simulating the ITSO Sock Store’s SAP based Sock Domain. We took you through the steps to configure your ITSO Sock Store’s SAP based Sock Domain complete with a Java based web service hosted on SAP. We then described how to publish the Sock web service into the SAP UDDI Repository by using the SAP Services Registry. By publishing the web service, you made it discoverable by WebSphere Service Registry and Repository. Before you could proceed, you needed to identify the exact access details needed to connect to SAP UDDI Repository. To gather these access details, you needed to retrieve both the UDDI API URLs and its accompanying...
security settings. By now, you have now created an analogue for the ITSO Sock Store's SAP based Sock Domain. This environment is used by importing the Sock web service metadata into the IBM based Retail Domain by way of WebSphere Service Registry and Repository and SFM.

**Setting up the eBusiness service domain**

Perform the steps in this section to install and set up SFM in the eBusiness domain.

**WebSphere Message Broker**

WebSphere Message Broker requires WebSphere MQ. You can install WebSphere Message Broker and WebSphere MQ using the launchpad, as described in the WebSphere Message Broker Information Center found at the following address:


You should also install the WebSphere Message Broker Toolkit.

For this paper, the default broker configuration is sufficient. To create this configuration, follow the WebSphere Message Broker Information Center instructions to create the default configuration. You can find these instructions at the following address:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/index.jsp?topic=/com.ibm.e tools.mft.doc/ae20200_.htm

**SFM domain**

To create the eBusiness domain shown in Figure 3 on page 8, you should complete the following steps, which come from *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686.

---

**Note:** *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 is referenced a number of times in this paper. Whenever you are following instructions from this additional paper, remember to change the domain names and all other variables to reflect the configuration of your system for this paper.

1. Install WebSphere Service Registry and Repository.
2. Install the WebSphere Service Registry and Repository Feature Pack for Service Federation Management.
3. SCMP enable WebSphere Service Registry and Repository.
   This task involves loading various items into the WebSphere Service Registry and Repository, creating a ServiceRegistry business object, and creating a required WebSphere Application Server variable.
4. SCMP enable WebSphere Message Broker.
   This task assumes that you have installed and configured WebSphere Message Broker (see "WebSphere Message Broker" on page 28).
5. Import the cert.arm file into WebSphere Application Server.
   Perform this step to load the cert.arm file you created as part of enabling WebSphere Message Broker with SCMP into the WebSphere Application Server hosting WebSphere Service Registry and Repository in the eBusiness domain.
6. Install the SFM Console.
   This task involves creating a database and data source, installing the SFM coordinator application, installing Business Space, and installing the SFM Business Space widgets. These tasks should all be done in the WebSphere Service Registry and Repository you installed in the eBusiness domain.

7. Locate servers in SFM.
   Within SFM, use these steps to create an alias and an alias-to-host mapping for the machine running the WebSphere Service Registry and Repository in the eBusiness domain. Then load the service document for the WebSphere Service Registry and Repository. Repeat these steps again for WebSphere Message Broker.

8. Create domains in SFM.
   Create the eBusiness domain in SFM. Add WebSphere Message Broker as the connectivity provider for the eBusiness domain.

9. Create a federation containing the domains.
   Create a new federation and to add just the eBusiness domain to it.

After completing all of these steps, you should have an infrastructure that matches Figure 3 on page 8. In the Sock domain, the SAP products should be installed, and the web services to be shared should be installed. In the eBusiness domain, the SFM Console is installed and working, the eBusiness domain is created in SFM, and a federation has been created.

Beginning the ITSO T-Shirt Store scenario

At this point, you should have installed and configured all of the prerequisite software and have an infrastructure that matches Figure 3 on page 8. The eBusiness domain is fully configured for SFM, and is ready for use by SFM in sharing web services. By completing the next few sections, you take the initial ITSO T-Shirt Store infrastructure for the Sock domain, add to it all of the required products, and complete all of the required enablement and configuration to be able to use SFM to share services from the SAP Application Server in the Sock domain to the eBusiness domain.

Installing WebSphere Service Registry and Repository

The Sock domain must have WebSphere Service Registry and Repository installed so that the domain can use it as a registry server and a domain server.

Perform the following steps:

1. Install WebSphere Service Registry and Repository, as documented in the WebSphere Service Registry and Repository Information Center found at the following address:

2. Using the PMT graphical tool, create a WebSphere Service Registry and Repository profile, as documented in the Information Center found at the following address:

   You should select the Typical profile when prompted, as this profile uses all of the default settings, including the built-in Derby database.
3. Load and activate the required configuration profile inside WebSphere Service Registry and Repository by following the procedure in the Information Center found at the following address:


For this paper, use the GovernanceEnablementProfile_v70.zip file.

**Installing WebSphere Enterprise Service Bus**

In order for proxies to be created in the Sock domain, a connectivity provider must be installed. In this paper, you install WebSphere Enterprise Service Bus as the connectivity provider for the Sock domain.

Perform the following steps:

1. Install WebSphere Enterprise Service Bus as documented in the WebSphere Enterprise Service Bus 7.0 Information Center found at the following address:


2. Using the PMT graphical tool, create a typical stand-alone server profile, as documented in the WebSphere Enterprise Service Bus 7.0 Information Center found at the following address:


You should use administrative security and provide a user name and password when prompted. Start the server after the profile is created.

**Installing the SFM feature pack**

The WebSphere Service Registry and Repository Feature Pack for Service Federation Management must be installed on the machine in the Sock domain that is hosting the WebSphere Service Registry and Repository that will be used as the domain server and the registry server.

To install the SFM feature pack, follow the instructions in the SFM section of the WebSphere Service Registry and Repository Information Center found at the following address:


When prompted, you should select both features to install.

After the installation has completed, the SFM feature pack files should be in the <WAS_HOME>\feature_packs\WSRR-SFM directory.

**SCMP enabling WebSphere Service Registry and Repository**

In order for a WebSphere Service Registry and Repository to be used as a domain server and a registry server, WebSphere Service Registry and Repository must be SCMP enabled.

The section “SCMP Enable a WebSphere Service Registry and Repository” in *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 contains detailed
steps for this task. This task involves loading various items into WebSphere Service Registry and Repository, creating a ServiceRegistry business object, and creating a required WebSphere Application Server variable.

**SCMP enabling WebSphere Enterprise Service Bus**

WebSphere Enterprise Service Bus V7.0 already contains the functionality required to support SCMP. However, some configuration is required. The section “SCMP Enable WebSphere Enterprise Service Bus” in *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 contains detailed steps covering this task.

After WebSphere Enterprise Service Bus is SCMP enabled, SFM is able to recognize it as a connectivity provider and use it to create proxies for the web services it shares.

**Additional security steps**

As ITSO T-Shirt Store is using WebSphere Application Server administrative security in the eBusiness domain, you must configure this security to work in both domains. There are two additional steps that must be completed to allow the various systems to work together. *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 contains detailed sections covering each of the following steps:

1. Share SSL certificates.

   In order for WebSphere Application Servers and WebSphere Enterprise Service Bus to communicate securely with each other, SSL certificates must be exchanged. In this paper, the WebSphere Application Server in the eBusiness domain that is hosting the SFM Console and WebSphere Service Registry and Repository, the WebSphere Application Servers hosting WebSphere Service Registry and Repository in the Sock domain, and the WebSphere Enterprise Service Bus must all have each other’s certificates.

2. Create and share a new LTPA token.

   In order for the two WebSphere Service Registry and Repository instances and WebSphere Enterprise Service Bus to communicate, they must share the same LTPA token. A new LTPA token must be generated in one system and exported before being imported into the other two systems.

**Making the SAP services available to SFM**

ITSO Sock Company needs to share its existing SAP based web services to the ITSO T-Shirt Company, which runs on an IBM based environment. This integration is brokered by WebSphere Service Registry and Repository to link to SAP NetWeaver UDDI Repository. WebSphere Service Registry and Repository has built-in functionality called UDDI Synchronization to achieve this integration. To connect to SAP, WebSphere Service Registry and Repository needs to be configured with the API URLs and policy settings.

**Note:** The procedure to derive the configuration file for the UDDI synchronization can make use of a Java based GUI tool or manually crafting the XML. For the purposes of the paper, we use the manual approach to build the XML.

These settings need to be entered into WebSphere Service Registry and Repository through an XML based configuration file. This configuration file allows for storage of the connection details to one or many different UDDI repositories. It abstracts the connection details to allow
for the use of the appropriate security mechanisms for a particular registry. At the same time, the configuration file also allows for the importation of a specific search criteria about which services are selected for import. However, for this paper, we focus only on how to integrate one particular SAP NetWeaver UDDI Registry and select all of its services.

Refer to the following address for details about more complicated scenarios:

Editing the UDDI synchronization configuration XML file

To edit the UDDI synchronization configuration XML file, perform the following steps:

1. Using the additional materials supplied with this paper (see “Additional material” on page 46), make a copy of and open the file called WSRR UDDI Config Template.xml. This file is a template that you can use to modify to suit your environment. Figure 27 shows this file.

   ![XML code snippet](image)

   **Figure 27** Configuration

2. Note the tag named SERVICE_CONTENT_LOCATION. Replace [%wsrr_host%]:[%port%] with the host name of your WebSphere Service Registry and Repository and the port with your SSL secured application host.

3. Place a name for your SAP System at the following position:
   WSRR_UDDI_CONFIGURATION -> UDDIREGISTRIES -> UDDIREGISTRY -> Alias
4. Now you see Inquiry, Publish, and Security URL tags. Update these tags to match the API URLs shown in Figure 27 on page 32. Replace all occurrences of [%sap_host%]:[%port%].

5. Review the Server Auth and UDDI Auth Token tags for Inquiry, Publish, and Security. Then update these fields the SAP UDDI API Access Policy in Figure 26 on page 27.

6. Update the user name and password fields to match the credentials used to access your SAP Application Server Java. After they are stored in WebSphere Service Registry and Repository, the passwords are hashed and obfuscated.

7. Replace [%sap_host%] with the host name of your SAP system in the URL_PATTERN tag.

8. Review the original template to ensure that you have populated the XML configuration file appropriately.

**Applying the UDDI synchronization XML file**

The configuration file for the UDDI synchronization is now complete and needs to be placed into WebSphere Service Registry and Repository.

Perform the following steps:

1. Open a web browser and navigate to the WebSphere Service Registry and Repository in your WebSphere Enterprise Service Bus domain:
   
   https://<wsrr_host>:<app_port>/ServiceRegistry

2. Log in using the credentials you created when creating the server profile.

3. Select the **Configuration** profile.
4. From the menus, select **Active Profile** → **UDDI** (Figure 28).

5. The page that opens should be titled UDDI Synchronization Configuration. Click **UDDI Synchronization configuration (UDDIConfiguration)** (Figure 29).
6. The UDDIConfiguration page contains a large text box with XML code in it. This XML code represents the existing UDDI configuration within WebSphere Service Registry and Repository. Select all the XML code in the text box and delete it. Paste in the XML code you created in the WSRR UDDI Config Template.xml file (Figure 30).

![WebSphere Service Registry and Repository Configuration Item](image)

Figure 30  Pasting in the XML code

7. Click OK.
8. The UDDI Synchronization Configuration page reopens. You should see a message that the UDDIConfiguration Configuration item was successfully updated (Figure 31).

![WebSphere, Service Registry and Repository](image)

**Figure 31** The confirmation message

### Starting a UDDI synchronization task manually

The configuration file is now in place within WebSphere Service Registry and Repository. Now you need to tell WebSphere Service Registry and Repository to start a job to retrieve the latest data from its known UDDI repositories. Normally, you would rely on a scheduled job to pick up these changes for you. However, for the purposes of this paper, we manually start this job by executing commands from the command line.

This manual process starts a process that is long running. It is possible that you could encounter some timeouts for the console or for the SOAP interface. These timeouts might occur on slow systems, when there a large number of UDDI repositories, or with a large number of entries in a single registry. If you encounter this issue, you need to increase the timeouts manually.

Refer to the following address if you would like more details about manually starting a scheduled task:


Perform the following steps:

1. Ensure that your WebSphere Service Registry and Repository server profile is started.
2. Identify your cell, node, and server names for your WebSphere Service Registry and Repository profile.
3. Determine the fully qualified path to your WebSphere Service Registry and Repository installation root.
4. Execute the following two commands:
   - `cd <WSRR_INSTALL_ROOT>\WSRR\admin\scripts_cell`
   - `\WSRR_INSTALL_ROOT\bin\wsadmin -f executeSchedulerTaskImmediate.jacl -cell <CELLNAME> -node <NODENAME> -server <SERVERNAME> -taskname UDDIScheduledTask`
Here are the populated commands:

```
D:\IBM\WebSphere\ServiceRegistry\admin\scripts_cell
D:\IBM\WebSphere\ServiceRegistry\bin\wsadmin -f
executeSchedulerTaskImmediate.jacl -cell parrott-betaNode02Cell -node
parrott-betaNode02 -server server1 -taskname UDDIScheduledTask
```

These commands start the bootstrap process to start the UDDIScheduledTask.

5. After a short time, a window opens and prompts you for your user name and password (Figure 32). Enter the user name and password and click **OK**.

![Figure 32 Login window](image)

A short time later, the process completes (Figure 33).

![Figure 33 The completed process](image)
Reviewing the imported SAP web service

The SAP based Sock web service should have had its metadata imported into WebSphere Service Registry and Repository. Verify that the Sock web service was imported by performing the following steps:

1. Open a web browser and navigate to the WebSphere Service Registry and Repository in your Sock domain:
   https://<wsrr_host>:<app_port>/ServiceRegistry
2. Log in using the credentials you created after creating the server profile.
3. Select the Administrator profile.
4. From the home page, in the Service Documents section, click WSDL Documents (Figure 34).

![Figure 34 Selecting WSDL Documents](image-url)
5. Click the WSDL document named `Sock?wsdl&mode=sap_wsdl` (Figure 35).

6. Review the metadata for the Sock web service that was imported into WebSphere Service Registry and Repository (Figure 36).
Using Service Federation Management to share services

Having completed all of the steps to this point, the Sock domain now has all of the products used by SFM installed, configured, and SFM enabled. The SAP services are in WebSphere Service Registry and Repository in the Sock domain, and are available to SFM.

Preparing the SFM domain and federation

Before you can share the services from the Sock domain to the eBusiness domain, the Sock domain must be created in the SFM Console and added to the federation. There are a number of steps that must be completed to achieve this task. For each step, you can find detailed instructions in *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686:

1. Create aliases.
   The user name and password used by SFM to communicate with WebSphere Service Registry and Repository and WebSphere Enterprise Service Bus in the Sock domain must be stored in SFM. This task is accomplished by creating aliases.

2. Create host to alias mapping.
   The aliases that were just created must be linked to the host names and port numbers of the machines with which they should be used.

3. Load the service documents.
   In order for SFM to know about the available registry server, domain server, and connectivity provider, the Service Documents for these components must be added in the SFM Console.

4. Create the Sock domain in SFM.
   A domain must be created in SFM for the Sock domain. This domain contains the registry server and the connectivity provider.

   **Note:** Remember that in this paper that each domain has its own SFM domain server, so the domain server should be created on the domain server in WebSphere Service Registry and Repository in the Sock domain.

5. Add the Sock domain to the existing federation.
   In order to share services between the Sock domain and the eBusiness domain, both domains must be in the same federation. The existing federation already contains the eBusiness domain, so the newly created Sock domain must also be added to the federation.

Sharing the services from the Sock domain to the eBusiness domain

To make managing large numbers of services easier, SFM does not work on individual services. It allows them to be grouped together into service groups, and then services are shared by sharing the service groups.
Creating a service group

Perform the following steps to create a service group containing the ITSO Sock Store web services that are to be shared:

1. Open a web browser, and navigate to the SFM Console. Use the following URL and change the host name and port for the host name and port used to access the WebSphere Service Registry and Repository:
   
   https://<hostname>:9443/mum/enabler

2. Log in using the same user name and password used for WebSphere Service Registry and Repository.

3. Click the Domain Management tab.

4. Select All servers from the Domain server drop-down menu, and select the Sock domain that you created from the Domain drop-down menu.

5. Click the Service groups tab.

6. Click the Create service group button. The window shown in Figure 37 opens.

   ![Figure 37 Creating a service group](image)

   Figure 37 Creating a service group

7. Enter a name in the Service group name field, for example, Order Services.

8. Set Visibility to Public and Sharability to Shareable.

9. Click Save.

10. Click the Add service endpoints button.

11. Check the check boxes next to the services you want to include in the service group. If you use the services provided with this paper, select the SockPort service endpoint (Figure 38).

   ![Figure 38 Adding the endpoint to the service group](image)

   Figure 38 Adding the endpoint to the service group

12. Click the Add selected service endpoints button.

You have now created a service group containing the services to be shared.
**Sharing the service group**

Now that the services in the Sock domain are contained within a service group, this service group can be shared from the Sock domain, to the eBusiness domain.

Perform the following steps:

1. In the SFM Console, click the **Federation Management** tab.
2. In the Federation server drop-down menu, select **All servers**.
3. In the Federation drop-down menu, select the federation you created, which contains the Sock and eBusiness domains. You should see the two boxes representing the two domains. You can drag these boxes around to arrange them however you want.
4. Hover the mouse over the Sock domain box. A small white circle should appear at the box’s bottom. Click that circle and drag it towards the box for the eBusiness domain. An arrow should appear. Release the mouse button when the mouse is over the eBusiness domain box. An arrow should be created between the two boxes (Figure 39).

![Figure 39 Sharing the services](image)

5. At the right, the Share Editor should open (Figure 40). In the drop-down menu for New share, select the service group you created, for example, **Order Services**, and click **Share**.

![Figure 40 Selecting the service group to share](image)
The share details information should appear within the Share editor (Figure 41).

6. If you want to enforce all requests using HTTPS, check the HTTPS check box. For this paper, we leave this box clear.

7. Select in the drop-down menu whether you want Validation capability on each proxy. For this paper, we leave this setting set to None.

8. Select which Proxy positions you require. In this paper, we use proxies on Both sides.

9. Click Create. SFM creates any required proxies, and moves the service information to the relevant registry servers, so that the shared services can be accessed from the eBusiness domain. The Create button will change to Creating share... and be disabled while the process is running. This process may take a number of minutes. If there is a problem, a window opens and displays an error.

10. After the share has successfully completed, the Creating share... button will change to Verify and repair.

Demonstrating and testing the end result

The web services in the service group in the Sock domain have been shared with the eBusiness domain, which means that the web services are now available to service consumers within the eBusiness domain, despite being hosted on the SAP Application Server in the Sock domain. We achieved this configuration using the WebSphere Service Registry and Repository UDDI Synchronization functionality to bring the WSDL documents into WebSphere Service Registry and Repository, and using the SFM Console, which reduced the complexity of performing the sharing. Figure 2 on page 6 show the final state for a particular service.

This final section of the paper shows what has been created in the SFM Console, the WebSphere Service Registry and Repository instances, WebSphere Enterprise Service Bus, and WebSphere Message Broker as a result of sharing these services. This setup allows you to verify that the share has worked successfully and also helps you understand how SFM actually makes the services available to the eBusiness domain.
The section “Demonstrating and testing the end” in *Strengthening Your ESB with WebSphere Service Registry and Repository*, REDP-4686 covers all of the following sections in more detail.

**SFM Console**

The first place to see what has been created is within the SFM Console itself. Previously, you created a service group within the Sock domain. In addition to this group, there should now be a second group for the proxy.

As the service group has been shared to the eBusiness domain, two service groups should exist in the eBusiness domain.

**WebSphere Service Registry and Repository in the Sock domain**

The registry server in a domain stores all of the metadata about the services being shared. When you originally created the service group containing the services to be shared, this metadata was stored in the registry server, which is WebSphere Service Registry and Repository.

Earlier, when you SCMP enabled WebSphere Service Registry and Repository in the Sock domain, one step involved creating a ServiceRegistry business object (called Sock Registry Server in our example). When the service group was created in the registry server, a relationship was created from the ServiceRegistry business object to the service group, and a relationship created from that group to the WSDLPorts. After the sharing, a second service group also exists for the proxy.

**WebSphere Service Registry and Repository in the eBusiness domain**

After the share has been successfully created, two service groups are created in the registry server in the eBusiness domain and have a relationship on the ServiceRegistry business object. One group contains the web services available to consumers within the eBusiness domain, which will be the web service for the proxy in WebSphere Message Broker. The other group contains the web service for the proxy in the Sock domain, which will be hosted in WebSphere Enterprise Service Bus.

**WebSphere Enterprise Service Bus**

In this paper, a proxy was created in the Sock domain. As we only have one connectivity provider in this domain, which is WebSphere Enterprise Service Bus, then this provider is where the proxy was created.

**WebSphere Message Broker**

In this paper, we created a proxy in the eBusiness domain. We configured WebSphere Message Broker as the connectivity provider in the eBusiness domain, so the proxy will be created in WebSphere Message Broker.
Calling the web service

To truly test that SFM has successfully shared the web services from the SAP Application Server in the Sock domain to the eBusiness domain, call one of the web services using the new endpoint in the eBusiness domain. This is the endpoint of the proxy, which in this paper is in WebSphere Message Broker in the eBusiness domain.

In the additional materials section of this paper (refer to “Additional material” on page 46), there is a web services client that allows you to call the Socks web service. The client EAR file is called WebServiceProjectEAR.ear.

The following steps show how to call the newly shared web service using the client. *Strengthening Your ESB with WebSphere Service Registry and Repository, REDP-4686* contains a more detailed description of these steps.

Perform the following steps:

1. Identify the endpoint of the web service in the proxy in the eBusiness domain by performing the following steps:
   a. Log in to the WSRR Console on the eBusiness domain.
   b. Select the Administration perspective from the drop-down menu at the top right. From the Home page, under Business Objects, select ServiceRegistry from the list.
   c. Select the ServiceRegistry business object eBusiness Registry Server.
   d. In the Relationships section, click the Service Group without the [S].
   e. In the Relationships section for the service group, under WSDLPorts, click the SockPort web service.
   g. Click the Content tab for the WSDL document to see the contents of the WSDL. The soap:address should be at the bottom of the WSDL file. This is the endpoint of the web service within the eBusiness domain. Make a note of this endpoint.

2. Install and start the client application on the WebSphere Application Server hosting WebSphere Service Registry and Repository in the eBusiness domain.

3. Load the client application in a web browser. The URL to access the client is as follows (change the host name and port number to those required to access WebSphere Service Registry and Repository):

   http://<hostname>:<port>/WebServiceProject/sampleSockProxy/TestClient.jsp

4. Click setEndpoint(java.lang.String) and, in the text box labelled endpoint, enter the endpoint for the web service in the eBusiness domain (the endpoint you found in step 1) and click the Invoke button.

5. Click getEndpoint() and click Invoke. Confirm that the endpoint that appears in the Result frame is correct.
6. Click the `orderItem(int)` operation. After clicking `orderItem(int)`, the client should look as shown in Figure 42.

```
Methods
- useNDI(boolean)
- getEndPoint()
- setEndPoint(java.lang.String)
- checkStock(int)
- backOrder(int)
- orderItem(int)
- getStock()

Inputs
- arg0: [blank]

Invoke Clear

Result
- result: N/A
```

**Figure 42**  *The web service client*

7. Enter an integer as the argument and click **Invoke** to call the web service.

This should result in a string being returned in the Result frame of the client application matching either:

- Success, your order has been placed.
- Failure, no socks are in stock.

When you see that this message returned, you have just invoked the web service that is hosted in the SAP Application Server in the Sock domain from the eBusiness domain. This means you have successfully used SFM to share services from the SAP Application Server in the Sock domain to the eBusiness domain.

In the ITSO T-Shirt Store scenario, any service consumers in the eBusiness domain now have access to the `checkStock` and `orderItem` Sock web services. They can easily look up the service in WebSphere Service Registry and Repository in the eBusiness domain, and then call the service at the endpoint referenced in the WSDL file.

**Additional material**

This section refers to additional material for this paper that can be downloaded from the Internet.

The material that is associated with this paper is available in softcopy on the Internet from the IBM Redbooks® web server. Point your web browser at:

```
ftp://www.redbooks.ibm.com/redbooks/REDP4685
```

Alternatively, you can go to the IBM Redbooks website at:

```
ibm.com/redbooks
```

Select **Additional materials** and open the directory that corresponds with the IBM paper form number, REDP4685.
The team who wrote this IBM Redpapers publication

This paper was produced at the International Technical Support Organization, Raleigh Center.

David Bell is the Integration Test Technical Lead for WebSphere Service Registry and Repository. He has 4 years of experience in SOA, primarily focused on WebSphere Service Registry and Repository. Prior to that experience, he spent 5 years working in the Java Technology Center. He has a First Class Honours Degree in Computer Science from the University of Nottingham. David is also an IBM Senior Inventor, a distinction that recognizes his contributions to invention within IBM.

David Parrott is a Technical Support Specialist for WebSphere Adapters. He specializes in supporting the SAP Adapter running on WebSphere Message Broker and WebSphere Process Server. David has worked for the WebSphere Adapter Support Team since joining IBM 3 years ago. Prior to joining IBM, David graduated from college with a Bachelors of Science in Computer Science with an emphasis in team-based software development.

Thanks to the following people for their contributions to this project:

- Martin Keen, IBM Redbooks Project Leader
- Michael Troxel, IBM SAP Basis Administrator
- Nick Butler, IBM STSM, Service Federation Management

Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at: ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on Facebook:
  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:
  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:
  http://www.linkedin.com/groups?home=&gid=2130806
> Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:


> Stay current on recent Redbooks publications with RSS Feeds:

http://www.redbooks.ibm.com/rss.html
Notices

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

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
This document REDP-4685-00 was created or updated on October 25, 2010.

Send us your comments in one of the following ways:

- Use the online Contact us review Redbooks form found at:
  [ibm.com/redbooks](http://ibm.com/redbooks)
- Send your comments in an email to:
  [redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at [http://www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

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

- IBM®
- Redbooks®
- Redpaper™
- Redpapers™
- Redbooks (logo) ®
- WebSphere®

The following terms are trademarks of other companies:

- ABAP, SAP NetWeaver, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.
- Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.