IBM® Data Studio Developer comes with a capability (called Data Web Services) which allows you to define Web services based on database operations. This procedure is driven by metadata and supported by tooling so that no code generation is required and no deployment artifacts need to be written by hand. Data Studio generates runtime artifacts for those services which can be deployed into an application server environment. Runtime artifacts for different environments can be generated since the service definition and public interface (WSDL) are abstract and completely independent from the service implementation. With IBM Data Studio Developer 1.2 (or later), the WebSphere® DataPower® XI50 Integration Appliance is supported as another runtime environment besides the traditional J2EE application servers. This tutorial shows you how to use Data Studio to generate Data Web service runtime artifacts for DataPower. Additionally, learn how those artifacts can be deployed on an IBM WebSphere DataPower XI50 Integration Appliance.

Before you start
About this tutorial
This tutorial gives you the basic steps of how to develop a Data Web Service with IBM Data Studio and shows you how to deploy it on an IBM WebSphere DataPower XI50 Integration Appliance.

Motivation
IBM Data Studio Developer provides an easy way to expose database data as a service. While the service definition and artifact generation are important to understand, if you want to get serious about SOA, you must consider the problems of security, performance, governance and monitoring. Those issues can many times not be solved with the service definition, but they require additional infrastructure and configuration steps. Even though it's possible to create an "enterprise-ready" SOA environment with traditional J2EE application servers, it might not always be the best approach for exposing enterprise data as services.

IBM WebSphere DataPower provides a distinct and competitive hardware, software, and infrastructure stack which allows you to address many of the SOA challenges mentioned earlier. Now, this infrastructure can be leveraged for Data Web Services with IBM Data Studio
Developer 1.2 (or later) supporting WebSphere DataPower XI50 Integration Appliance as runtime environment and the DataPower enhancements for database access in the 3.7.1 firmware release.

Using WebSphere DataPower XI50 Integration Appliance as the hosting environment for Data Web Services allows you to leverage the superior support of network protocols. This gives a wide variety of clients the ability to talk to DB2 — without even being database-aware. Furthermore, all the other DataPower advantages like security, XML hardware acceleration for parsing, schema validation and XSLT processing can be used.

Objectives
In this tutorial, you learn what steps are necessary to create Data Web Service runtime artifacts for the WebSphere DataPower XI50 Integration Appliance with IBM Data Studio Developer 2.1. Furthermore, see how to create a DB2 data source on DataPower, how to transfer the service artifacts from Data Studio to DataPower, how to create an HTTP-based binding configuration using the XSL Accelerator, and how to create the WSDL/SOAP-based service binding using the Web Service Proxy.

Prerequisites
This tutorial is written for users who have basic knowledge of Web Services, databases, IBM Data Studio and IBM WebSphere DataPower Appliances.

System requirements
To run the examples in this tutorial, you need IBM Data Studio Developer 1.2 or 2.1 as well as DB2 version 8 or higher with the sample database. For service deployment, you need an IBM WebSphere DataPower Integration Appliance XI50 with firmware level 3.7.1 and the ODBC package.

Overview
In the tutorial you will:

1. Create a Data Web service and DataPower runtime artifacts with IBM Data Studio Developer
2. Create a DB2 data source on DataPower
3. Upload the generated service artifacts to DataPower
4. Configure an XSL Accelerator for HTTP POST XML binding
5. Configure an XSL Accelerator for HTTP GET binding
6. Configure a WS-Proxy for SOAP over HTTP binding

How it works
Introduction to IBM WebSphere DataPower SOA appliances
IBM WebSphere DataPower SOA Appliances are purpose-built, easy-to-deploy network devices that simplify, secure, and accelerate your XML and Web services deployments while extending your SOA infrastructure. These appliances offer an innovative, pragmatic approach to harness the power of SOA. By using them, you can simultaneously use the value of your existing application, security, and networking infrastructure investments.
A DataPower appliance can take on different roles in an SOA environment. The appliance can act as a simple XSL accelerator and transformation engine, be used as a Firewall and Security device up to a Multi-Protocol Gateway, and even function as an Enterprise Service Bus (ESB).

This tutorial focuses on DataPower as a transformation and protocol conversation engine to transform Web service requests into database calls and the result back into service responses to implement a data access service. Figure 1 shows this transformation process:

**Figure 1. Transforming requests into database calls**

A transformation is defined by a set of rules which determine how an input is mapped to an output. A very common format for Web service is XML. XSL is a way to define such kind of mapping rules for XML-based inputs. DataPower provides first class support for XSL processing with its purpose-built XML and XSL processing stack. Therefore, XSL is used to transform service requests into database calls and back. DataPower provides the XSL `<dp:sql-execute>` extension element to perform the database access operations.

To learn more about the IBM WebSphere DataPower appliance family check out the *IBM WebSphere DataPower SOA Appliances Part 1* (see Related topics).

**Use IBM Data Studio Developer to generate DataPower service artifacts**

While the information provided so far gives you the ability to write the appropriate XSL scripts by hand to implement the transformation mappings necessary for a data access service, this is a very time-consuming and error-prone task. IBM Data Studio Developer provides a quick and easy way to expose database operations as Web services. After a service has been defined, artifacts for a specific runtime environment can generated. Traditionally, those are artifacts for a J2EE/JavaEE-based application server. But now it's possible to also generate service artifacts for DataPower. The generated artifacts consist of the WSDL file of the service as well as a set of XSL scripts defining the mapping between service requests and database calls. Those artifacts can now be deployed on a DataPower XI50 appliance. Figure 2 shows how this process works:

**Figure 2. Generate DataPower artifacts with Data Studio Developer**
Later, this tutorial uses a simple example to show you how to use Data Studio Developer to create a Data Web Service, how to generate the artifacts for DataPower, and how to deploy the service on a DataPower XI50 appliance.

**Deploy the Web service**

After the artifacts have been generated, it's time to deploy them at the DataPower appliance. DataPower provides different configuration categories for different kind of services as Figure 3 illustrates.

**Figure 3. Various service configurations available on DataPower**

For simplicity, this tutorial only uses the XSL Accelerator to implement HTTP-based bindings and the Web Service Proxy to implement the WSDL/SOAP-based binding for the service. The generated artifacts can also be used with different configurations, for example, using MQ as message transport, or putting additional security like authentication/authorization or secure communication via SSL around it. The most common way to configure a DataPower appliance is through the WebGUI Web interface. All descriptions and screen shots in this tutorial are based on the DataPower WebGUI.

**Configuration in DataPower**

DataPower has a completely metadata-driven configuration approach. All service configurations have a front-end and back-end configuration as well as a processing policy which defines one or more processing rules. Depending on the selected service configuration, some of the configurations might be masked. For example, an XSL Accelerator configuration does not have an explicit back-end setup.

**Figure 4. DataPower configuration**

**Front end**

The *front end* defines the "client-facing" part of a configuration. It's main purpose is to configure protocol-specific settings like TCP/IP port, encryption, transport protocol versions, properties, and the like.
Back end

The back end defines the configuration of the service and server the processed request should be sent to. It usually contains protocol-specific settings and addresses information of the back-end service.

Note: A database is not considered to be a back-end system for DataPower. In case of Data Web services, the requests are terminated at the DataPower appliance and not sent to a back-end system. Instead of providing a real back-end system with the configurations, we use a loopback configuration which turns an incoming request into a response. The database access happens inside XSL scripts using the <dp:sql-execute> extension element.

Policy

A policy defines a set of one or more processing rules. Every service configuration on DataPower contains a policy setup.

Rule

A rule defines a sequence of actions (like matching, validation, transformation, routing, and so on) which get applied to a request or response message. For example, the XSL artifacts generated by Data Studio are added to such a processing rule in form of a transform action.

Create a simple Web service and generate DataPower artifacts

This section shows how easy it is to create a simple Data Web Service for the DB2 sample database using IBM Data Studio Developer 2.1. You'll create a Web service containing a SELECT, UPDATE, INSERT and DELETE statement as well as a stored procedure call.

Step 1. Set up the Data Development Project

First, create a new data development project in Data Studio and name it DataPowerSamples. Base this project on a connection to a DB2 sample database. Your DataPower XI50 appliance also needs to be able to connect to this database. The DataPower setup is described later in this document.

Figure 5. The data development project

Ensure that you are using the Data Project Explorer in the Data perspective in order to see the SQL Scripts and Web Services folders.

Step 2. Create SQL statements

Now it's time to develop the SQL statements that you'll expose as Web Service operations.
Operation 1: getEmployeeById (SQL SELECT)

The first statement is a simple SQL SELECT to retrieve an employee record from the employee table by a given ID.

To create a statement, right-click on the **SQL Scripts** folder and select **New -> SQL or XQuery** script.

**Figure 6. New SQL or XQuery statement**

![New SQL or XQuery Script](image)

Enter `getEmployeeById` as the statement name and keep the SQL editor setting. Click **Finish**. Alternatively, you can also use the SQL builder to assemble the statement.

Enter the following statement string in the editor window and save the statement:

```
SELECT * FROM EMPLOYEE WHERE EMPLOYEE.EMPNO = :empno
```

Now the statement should appear under the **SQL Scripts** folder like this:

**Figure 7. SQL Scripts folder with SELECT statement**

Operation 2: updateEmployeeSalary (SQL UPDATE)

Repeat the same step as shown in the `getEmployeeById` example but use the name `updateEmployeeSalary` and the following SQL statement:

```
UPDATE EMPLOYEE SET SALARY = :newSalary WHERE EMPNO = :empno
```
Operation 3: insertEmployee (SQL INSERT)
Repeat the same step as shown in the `getEmployeeById` example but use the name `insertEmployee` and the following SQL statement:

```sql
INSERT INTO EMPLOYEE
VALUES (:empno, :firstName, :midInitial, :lastName, :workDepartment, :phone, :hireDate, :job, :educationLevel, :sex, :birthDate, :salary, :bonus, :commission)
```

After creating all the statements, the SQL Scripts folder should look similar to the one in Figure 8.

Figure 8. SQL Scripts folder with all statements

Operation 4: BONUS_INCREASE (stored procedure)
The DB2 sample database already comes with the BONUS_INCREASE stored procedure. You can locate the procedure by navigating to the Stored Procedure folder under the appropriate schema in the database connection in the Data Source Explorer view:

Figure 9. Data Source Explorer with BONUS_INCREASE stored procedure

Now you can start assembling the Web service.

Step 3. Create the Web service
Create a new Web service by selecting the DataPowerSamples project and right-clicking on the Web services folder. Select New Web Service ....

In the New Web Service dialog you can specify a Web service name and a namespace URI. In this instance, use DataPowerSampleService as the Web service name and `http://ibm.com/example/DataPowerDWS` as the namespace URI as Figure 10 illustrates.
Figure 10. New Web service

Click Finish to create the service.

To add the SQL statements as Web service operations to your service, simply drag and drop them from the SQL Scripts folder on to the DataPowerSampleService Web service:

Figure 11. Drag and drop SQL statements

To add the BONUS_INCREASE stored procedure to the Web service, drag and drop it from the Stored Procedures folder into the Web Service:
Figure 12. Drag and drop the BONUS_INCREASE stored procedure

Now all the operations are added to the service, and you can do some fine tuning.

Select **DataPowerSampleService** and double-click on the **getEmployeeById** operation to bring up the Edit Operation dialog. Check the **Fetch only single row for queries** option since you know that this query only returns one row at a time - an employee record - and click **Finish**. This setting simplifies the XML structure of the service response.
The BONUS_INCREASE procedure returns a result set. Since the DB2 catalog does not contain result set metadata for stored procedures, the generated XML schema for the response message will, by default, contain only a very generic schema definition using the <xsd:any> element. The Data Web Services tooling lets you adjust the result set schema by executing the procedure and taking the schema information from the returned result set metadata. To do that, double-click on the BONUS_INCREASE operation in the DataPowerSampleService to bring up the Edit Operation dialog. Click Next to get to the Generate XML Schema for Stored Procedure dialog and click Generate.

Since the BONUS_INCREASE procedure has input parameters, a new dialog appears where you can specify input values. Specify 1.1 as P_BONUSFACTOR and 100000 as P_BONUSMAXSUMFORDEPT and click OK. This executes the procedure, and the metadata
from the returned result set can be analyzed. After the execution, you can click **Finish** on the Edit Operation dialog. That concludes the fine tuning of the Web service operations.

**Figure 15. Execute stored procedure**

![Specify Parameter Values - BONUS_INCREASE](image)

**Step 4. Build service artifacts**

Up to this point, the tutorial has not discussed the difference between a service to be deployed in a J2EE environment or one to be deployed on DataPower. In this step, you will create the service runtime artifacts, so you have to decide which runtime environment to use.

Right-click on the **DataPowerSampleService** and select **Build and Deploy ...**

Select **DataPower** as the Web server type. Ensure that REST and SOAP bindings are selected and provide a data source name for the artifact.dataSource property in the Parameters list. The name needs to match the data source, which you will set up on DataPower to connect to the DB2 sample database. The DB2 data source setup on DataPower is shown in the next section. Click **Finish** to generate the artifacts.
The artifacts are written to a sub-directory inside the Data Development project. Since the Data perspective does not show the created folder, you need to open the Navigator view as shown:

**Figure 16. Deploy Web service**

![Deploy Web Service dialog box](image)

**Figure 17. Navigator view**

![Navigator view](image)
The DataPowerSamples project contains a DataServerWebServices folder. In there you will find one directory for every Web service defined in your Data Development Project. A Web service folder contains the service metadata information in the .metadata folder. In case DataPower was selected as the runtime environment, you will also find an artifacts folder. This folder contains all the necessary service implementation files which need to be copied to DataPower appliance.

**Note:** If you don't see the artifacts folder, you need to refresh the project by right-clicking the DataPowerSamples project and select **Refresh** from the menu as Figure 18 illustrates.

**Figure 18. Refresh project**

This concludes the work in Data Studio. Now it's time to take care of the DataPower configuration.

**Set up a DB2 data source on DataPower**

This section shows you how to set up a DB2 data source on DataPower using the DataPower WebGUI.

1. Log into the DataPower WebGUI using your preferred domain.
2. To create or modify an SQL data source, expand **NETWORK** in the left-hand menu and click on **SQL Data Source**. Click **Add** to create a new SQL data source as Figure 19 indicates.
3. In the next dialog, provide all the necessary connection information. For the Name, enter `DB2LUW95`. Make sure the Admin State is set to `enabled` or DataPower will not connect to the database. Select `DB2 (version 9)` as the Database type. The version 9 driver also works with older DB2 versions. Enter your connection username and password. Enter the Data Source ID (sample), the Data Source Host (9.48.109.209), and Data Source Port (50000). Set Limit Returned Data and Allow Red-only Access to `off`. Set the Maximum Connections to 10.
4. Click **Apply** to save your settings. To persist configuration settings beyond DataPower shutdowns, click the **Save Config** button in the upper right corner.

5. The data source will be up and running (and ready for use) when the Op-State says **up** (Figure 21). You can refresh the view by clicking the Refresh List link -- it can take some time until the data source is connected to the database.

**Figure 21. Operation state is up**

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Op-State</th>
<th>Logs</th>
<th>Admin State</th>
<th>Database type</th>
<th>Data Source Host</th>
<th>Data Source Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2UW9S5</td>
<td>new</td>
<td>up</td>
<td></td>
<td>enabled</td>
<td>DB2v9</td>
<td>9.40.109.209</td>
<td>50000</td>
</tr>
<tr>
<td>DBTPW9S2</td>
<td>saved</td>
<td>up</td>
<td></td>
<td>enabled</td>
<td>DB2v9</td>
<td>9.39.129.210</td>
<td>50000</td>
</tr>
</tbody>
</table>

**Set additional driver parameters**

The Data Source Configuration Parameters tab allows you to specify additional connection parameters. In DB2, you can define the CLI/ODBC configuration keywords here. More information on the DB2 CLI/ODBC configuration keywords can be found in [The DB2 9.5 Information Center](https://www.ibm.com).
This concludes the data source setup. Now you need to copy the generated artifacts over to the DataPower box.

**Copy artifacts to DataPower**

In order to keep the setup simple, copy all artifacts into the local DataPower file system. DataPower also supports remote repositories like IBM's WebSphere Service Registry and Repository (WSRR).

**Create a Directory**

Since every service consists of multiple files, it's recommended to create a sub-directory for every service. Log into the DataPower WebGUI using your preferred domain. Select **Files and Administration > File Management** as shown below.

**Figure 22. File Management icon**

Click on the **Actions...** link next to the local: directory and select **Create Subdirectory**.

**Figure 23. Create subdirectory**

Provide the name "DataPowerSampleService" as the subdirectory name and click **Confirm Create** to create the directory.
Figure 24. Provide new directory name

Finally click Continue to get back to the directory overview.

Copy files

Expand the local:/DataPowerSampleService directory and select the Action... link next to the DataPowerSampleService directory. Click the Upload Files

Figure 25. Upload files

Browse to the Web services' artifacts directory inside the Data Studio workspace and select each of the generated files. To add a new file to the upload list, click the Add button. After all files are selected, click Upload to transfer them to the DataPower box.

Figure 26. Upload generated files

Finally click Continue to get back to the directory overview.

Now that you understand how to copy the artifacts, you can see how to configure the service.
Configure and Test an XSL accelerator for HTTP POST XML binding

This step shows how to configure a simple XSL Accelerator to implement the HTTP POST (XML) binding for the DataPowerSampleService. This binding takes XML request messages following the request message schema as defined in the WSDL document. The message format is very similar to the SOAP message except that there are no wrapping SOAP body and SOAP envelope tags around the XML - just the plain message payload.

The XSL Accelerator set up

1. Login to the DataPower WebGUI and click on the XSL Accelerator symbol:

   ![Figure 27. XSL Accelerator]

2. Click on the Add Wizard button
3. Select XSL Proxy Service and click Next (Figure 28).

   ![Figure 28. XSL Proxy Service]

4. Give the XSL Proxy the name DataPowerSampleService and click Next.

   ![Figure 29. Name the Proxy Service]

5. Select loopback-proxy as proxy type and click Next.

   ![Figure 30. Loopback Proxy Service]

6. On the next screen, keep 0.0.0.0 as the device address. Also, keep the provided device port (this is the TCP/IP port the request will later be sent to - this port needs to be remembered). Click Next.
7. Select the DataPowerSampleService_xml.xsl file from the store. Click **Next**.

**Figure 32. Select XSL file**

8. Verify your settings, and click **Commit** to create the XSL Proxy.

**Figure 33. Confirm XSL Proxy settings**

9. Click **Commit**.

10. Click on **View Policy** to modify the policy.

**Figure 34**

---

11. If you want to re-use this XSL Accelerator for multiple bindings, you have to tune the URL matching rule. Double-Click on the URL matching rule symbol as Figure 35 shows.
12. Create a new matching rule by clicking on the "+" button: Figure 36. Create New Match Rule

13. Give the rule the name `DataPowerSampleService_POST_XML`. Click on the Matching Rule tab, select Add and enter the URL pattern `/DataPowerSampleService/postXml/*`. Figure 37. Define URL pattern

Click Done to create the new matching rule.

14. Click the Apply Policy button in the policy window and close the policy window.

Now a client can send an HTTP POST request with an XML request message to DataPower to trigger the execution of a Web service operation. The HTTP POST binding requires to send an XML request message to the service. Data Studio Developer can be used to generate an appropriate XML request message for a Data Web service operation.

**Generate an XML request message with Data Studio Developer**

The Data Web services tooling allows you to generate an XML schema document for every operation. The XSD document contains the XML schema description for the operation's input and output message.

1. Right-click on the Web service operation and select Manage XSLT... from the context menu.
Figure 38. Select Manage XSLT...

2. Click the **Generate...** button in the **Configure XSL Transformations** dialog.

Figure 39. Generate the XML schema

3. This brings up the Save As dialog. You may change the suggested file name and location, but it’s recommended to keep it as is. Hit **Save** to save the XML schema to the project. And hit **Finish** in the Configure XSL Transformations dialog.
4. Right-click on your **DataPowerSamples** project and select **Refresh** from the context menu.

**Figure 41. Refresh the project**

5. The generated XSD file should now appear inside the **XML Schema** folder of the project.
6. With the generated XML schema, you can now generate an XML instance document representing the service request message. Right-click on the XML schema file and select **Generate -> XML File ...** from the context menu as Figure 43 shows.
Figure 43. Generate XML instance document from schema

7. Provide a file name and location for the XML instance file to be generated. Here you can keep the defaults. Click **Next**.
Figure 44. Specify name and location of XML request document

8. Select `getEmployeeById` as the XML Schema element you want to generate the XML instance document for. Data Web Services uses the operation name as the request message root name. Hit **Finish**.
Figure 45. Select the XML root element to generate the XML instance document for

9. Provide a valid value for the empno tag and save the file.

Figure 46. Final XML request message document

Test the HTTP POST XML Binding using cURL

cURL

cURL is a freeware program that is used widely to send HTTP requests from the command line. We use this program to test our examples. To download cURL, go to the cURL Web site at the following address: http://curl.haxx.se
You can use cURL to test the service. CURL is a command line tool and needs to be run from a terminal or DOS command window. The appropriate cURL command to invoke the service would look like this:

```
curl -X POST -H "Content-Type: text/xml" -v
  -d @DataPowerSampleService.getEmployeeById.default.xml
  http://DataPower:2058/DataPowerSampleService/postXml/getEmployeeById
```

A short explanation of the used cURL parameters:

- `-X` defines the HTTP method -- in this instance, use POST
- `-H` allows you to set or add HTTP header fields -- in this instance, add the Content-Type header to define text/xml as the request message type
- `-d` specifies the data to be sent with the request -- here, define the XML request instance document you generated earlier with Data Studio Developer
- `-v` turns on the verbose output

A short explanation of the request URL structure:

- `http://` defines the request protocol -- in this case, it's HTTP
- `DataPower` defines the host name or TCP/IP address of the DataPower appliance -- that value needs to be adjusted according to your environment
- `2058` the TCP/IP port you configured with the XSL Accelerator earlier
- `/DataPowerSampleService/postXml/` matches the URL pattern with defined in the processing rule
- `getEmployeeById` the XSL script uses this part of the URL to determine which operation to be executed

Here is the sample request message:

**Listing 1. Sample request message**

```
POST /DataPowerSampleService/postXml/getEmployeeById HTTP/1.1
User-Agent: curl/7.18.0 (i386-pc-win32) libcurl/7.18.0 OpenSSL/0.9.8g zlib/1.2.3
Host: 9.30.9.160:2058
Accept: */*
Content-Type: text/xml
Content-Length: 314

<?xml version="1.0" encoding="UTF-8"?>
<tns:getEmployeeById xmlns:tns="http://ibm.com/example/DataPowerDWS">
  <empno>000130</empno>
</tns:getEmployeeById>
```

Here the sample response message:

**Listing 2. Sample response message**

```
HTTP/1.1 200 Good
User-Agent: curl/7.18.0 (i386-pc-win32) libcurl/7.18.0 OpenSSL/0.9.8g zlib/1.2.3
Host: 9.30.9.160:2058
Content-Type: text/xml
```
Configure and test an XSL Accelerator for HTTP POST/GET binding (URL-encoded parameter)

Now it's time to implement the bindings for non-XML requests. In this case, the parameters are passed in as URL-encoded parameter/value pairs.

You can re-use the XSL Accelerator that you setup previously. All you need to do is add a new processing rule to the policy.

The XSL Accelerator set up

1. Login to the DataPower WebGUI and click on the XSL Accelerator symbol:

   **Figure 47. XSL Accelerator**

2. Click on the **DataPowerSampleService** XSL Accelerator:
3. Select the "..." button next to the Proxy Policy to open the policy:

**Figure 49. Modify the Proxy Policy**

4. Select the New Rule button to create a new rule for the URL-encoded bindings.

**Figure 50. Create a new rule**

5. Double-click on the URL pattern matcher icon and create a new matching rule - the same way as described in the HTTP POST XML binding setup before - called DataPowerSampleService_URL_encoded using /DataPowerSampleService/urlEncoded/* as URL pattern and save the new matching rule.

**Figure 51. URL pattern match icon**

6. Drag and drop a Transform action onto the flow:

**Figure 52. Drag and drop a transform action onto the flow**

7. Double-click it to open the configuration Window and select DataPowerSampleService_http_get_post.xsl as the processing control file:
8. Click **Done** at the bottom of the page.
9. Now you need to turn the URL-encoded parameter list into an XML document. To do that, drag and drop an **Advanced** action onto the flow between the URL pattern matcher and the transform action as Figure 54 shows.

**Figure 54. Add an Advanced action to the flow**

10. Open the advanced action by double-clicking on the symbol. Select **Convert Query Params to XML** as the action type and click **Next**

**Figure 55. Select "Convert Query Params to XML" as action**

11. Simply click **Done** in the next dialog. Your policy rule should now look like the one in Figure 56.

**Figure 56. Finished policy rule configuration**
12. Click **Apply Policy** and close the policy window. Click also **Apply** on the XSL accelerator configuration dialog to persist the changes. Select **Review changes, Save Config**, and then **Close**.

**Test the HTTP GET binding**

Now the URL-encoded bindings are available. A client can either use an HTTP GET request and provide the input parameters in the URL query string, or an HTTP POST request can be used where the MIME-Type is application/x-www-form-urlencoded.

For example, to call the `getEmployeeById` operation via HTTP GET, the URL would look like this:

```
```

You can now use the `/DataPowerSampleService/urlEncoded/*` URL pattern to activate the `DataPowerSampleService_URL_encoded` processing rule. Since HTTP GET has no meaning of a request message, you attach the input parameter as URL-encoded key value pairs in the URL query string.

Here is the sample response in a Web browser:

![Figure 57. Response message for HTTP GET binding in a Web browser](image)

**Configuring and Testing a Web Service Proxy for SOAP over HTTP binding**

Now you can configure the SOAP binding using a Web Service Proxy configuration. The SOAP binding setup requires an XML Firewall Loopback Proxy setup on the machine. The DataPower appliance only requires one of those. This tutorial describes the firewall loopback configuration as well before talking about the Web Service Proxy setup.

**Step 1. Configure an XML firewall loopback proxy**

The following steps describe the setup of an XML Loopback Firewall which will later be required when configuring the SOAP binding for the Data Web service.
1. Login to the DataPower WebGUI and click on the XML Firewall symbol:

   Figure 58. XML Firewall icon

2. Click the Add Wizard button and select Pass Thru as the Firewall type. Click Next.
3. Give the new firewall a name - like MyLoopbackFirewall - and click Next as Figure 59 shows:

   Figure 59. Naming the XML Firewall

4. Select loopback-proxy as the firewall type and click Next:

   Figure 60. Select loopback-proxy

5. Keep 0.0.0.0 as the device address, and choose an available TCP/IP port (the dialog usually provides the next available one automatically). Keep SSL disabled. Click Next:

   Figure 61. Keep default settings

6. Confirm your settings and click Commit:
7. The firewall is getting created and you will be prompted with a status page. Click **Done** to get to the XML firewall overview page.

8. Click on the newly created firewall to see the "Configure XML Firewall" page.

9. The default Request Type is SOAP. You may want to change it to **unprocessed** so that the firewall can handle any HTTP message - otherwise an HTTP 500 is returned if the request does not comply to the configured request type. Click on the firewall to open the configuration dialog. Set the Request Type to **Pass-Thru**.
Step 2. Configure the Web Service proxy

Now let's proceed with the SOAP over HTTP binding using the Web Service proxy.

1. In the Control Panel overview, click on the Web Service Proxy symbol:

   ![Figure 64. The Web Service proxy icon](image)

2. Select Add to create a new Web Service proxy.
3. Give the Web Service Proxy the name DataPowerSampleService and click on the Create Web Service Proxy button.

   ![Figure 65. Name the Web Service proxy](image)

4. Use the DataPowerSampleService_dp.wsdl file for the WSDL File URL. Select off under Use WS-Policy References and click Next.
5. The next dialog requires multiple configuration steps. DataPower parsed the WSDL document and provides us with a list of endpoints defined in the WSDL. For this exercise we only care about the SOAP (over HTTP) endpoint.
   1. To activate the SOAP endpoint, you need to configure a local endpoint handler, a URI and the binding.
   2. To assign an endpoint handler, you can either select an existing service handler from the drop-down box or create a new one by clicking the + button. When creating a new service handler, choose the **HTTP Front Side Handler** option (see Figure 67).

**Figure 67. Create a new Front Side Handler**

3. A new dialog window opens where you can specify the characteristics of the handler. Give the handler a name (like DataPowerSampleServiceFSH). You do at least need the support of the HTTP POST method to support SOAP over HTTP. The TCP/IP port used in the handler configuration will be used for the endpoint URLs later on (in our case 4444).
4. The last settings to be made on the Local interface of the SOAP endpoint is the endpoint URI and the SOAP version. You can keep the defaults (URI: /<ServiceName>/soapEndpoint; SOAP version: 1.1). Click the + Add link to add your local endpoint handler configuration.

5. Now you need to configure the Remote interface for the SOAP endpoint. Since DataPower is supposed to act as a Web service endpoint rather than forwarding the request to another server, you have to point to the local XML Loopback firewall which you created before. The protocol is HTTP, the IP Address is the loopback device (127.0.0.1), the port is 2050 (the one we configured for the loopback proxy). The endpoint URL doesn't matter -- you can keep it as is.

6. Click Next and Save Config in the upper right to persist the settings. This concludes the basic Web Service configuration. Now you need to take care of the implementation for the Web service.

6. To configure the service implementation, click on the Policy tab. That gives you a view of the WSDL structure. It allows you to apply rules on different levels. You will add two rules on the HTTP SOAP port level.
Figure 70. Add processing rule

7. Expand SOAP over HTTP port. You should add a rule for this port which implements the logic for all operations defined in that Web service. This rule contains an XSLT document (generated by Data Studio) which does implement the access to DB2:

1. Click on + Add Rule at the port level. This opens up a new dialog where you can specify the rule. Change the Rule Direction to Client to Server. Ensure that the match Action (which is the little black square with the equal sign) does match all incoming URLs. By hovering over it, you should see URL: *:

Figure 71. URL pattern match rule

2. Drag and drop the Transform icon onto the flow line. This adds an XSLT transformer to the process which implements the database access. Double-click on the icon to modify the transformation implementation. Upload the DataPowerSampleService_soap.xsl file which implements the SOAP binding for the Web service and assign it to this transform action. Click Done.

Create and deploy Data Web Services on WebSphere DataPower XI50 Integration Appliance
3. Finally, select the **Apply** button to conclude the Web service configuration. Figure 73 shows the policy rule definition for the Web service:

**Figure 73. Final policy setup**

This concludes the Web Service configuration.

**Test the SOAP over the HTTP binding**

A SOAP request can now be sent to the following endpoint URL. The deployed WSDL document can be requested using the following URL:


The SOAP endpoint URL is:

http://datapower:4444/DataPowerSampleService/soapEndpoint
You can use the WSDL URL with the Web Service Explorer in Data Studio Developer by performing the following steps:

1. Open the JavaEE perspective by selecting **Window -> Open Perspective -> Other...** from the top menu.

   **Figure 74. Open the JavaEE Perspective**

   ![Open JavaEE Perspective](image)

2. Start the Web Service Explorer tool by selecting **Run -> Launch the Web Service Explorer** from the top menu.
3. You will see a set of small icons in the upper right corner of the Web Service Explorer top menu bar. Click on the 2nd one from the right which says **WSDL Page** when you hover over it.

**Figure 76. Open the WSDL Main page in the Web Service Explorer**

4. Select the **WSDL Main** option in the left Navigator pane, enter the WSDL URL in the input field on the right pane, and click **OK**. This loads the WSDL file into the Web Service Explorer.

**Figure 77. Load WSDL file**
5. If the WSDL was loaded successfully, a new entry will appear in the left hand Navigator pane under WSDL Main. Expand the entry and drill down to any of the Web service operations listed there. Select one of the operations and provide the appropriate input values in the right hand pane. Click **OK** to execute the Web service. You will see the Web service response displayed in the lower right pane in case the Web service runs successfully.

**Figure 78. Test the SOAP over HTTP binding with the Web Service Explorer**

**Conclusion**

In this tutorial you learned how to use IBM Data Studio Developer to create a Data Web service and generate runtime artifacts for the WebSphere DataPower XI50 Integration Appliance. Furthermore, you learned how to configure a DB2 data source on the WebSphere DataPower XI50 Integration Appliance, to upload the generated artifacts and configure the different service bindings for the Data Web service. Finally you used simple clients - like a Web browser, CURL or the Web Services Explorer - to test the different service bindings.

You can find more information on IBM WebSphere Data Power, IBM Data Studio Developer, and Data Web services under the Resources section of this tutorial in case you want to take a deeper dive into the products used here.
Related topics

- The Redbook "IBM WebSphere DataPower SOA Appliances Part 1" gives you an overview of WebSphere DataPower SOA Appliances and helps you get started.
- Download a free trial version of IBM Data Studio Developer.
- The article, "Using WebSphere DataPower SOA Appliances to enable the Information as a Service pattern" has more details on how to implement database access on IBM WebSphere DataPower using the according XSL function and extension element.
- The Redbook "DB2 9 for z/OS: Deploying SOA Solutions" has more details on how to leverage IBM Data Studio Developer and IBM WebSphere DataPower XI50 Integration Appliance together with DB2 in an SOA environment.
- The article "Data Web Services: Build Web services the new way to access IBM database servers" gives you an introduction into Data Web services with IBM Data Studio Developer.
- "Using WebSphere DataPower SOA Appliances to enable the Information as a Service pattern" (developerWorks, Dec 2008): Learn more about using WebSphere DataPower SOA Appliances to accelerate Web service deployment within an SOA.
- The DB2 9.5 Information Center
- The Redpaper series on IBM WebSphere DataPower SOA Appliances provides a good introduction on IBM's SOA appliances and their usage.
- Download IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

© Copyright IBM Corporation 2009
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)