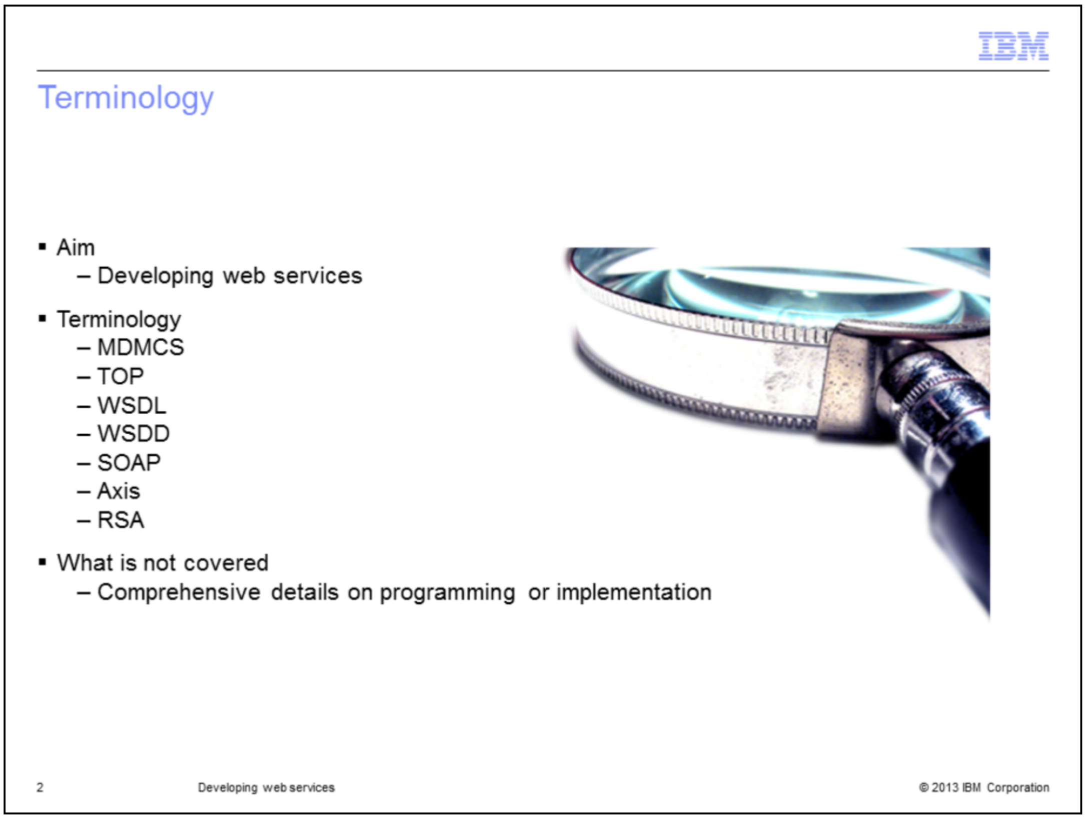
InfoSphere Master Data Management Collaboration Server V10

Developing web services

The topic of this presentation is how to use web services to integrate IBM InfoSphere® Master Data Management Collaboration Server version 10 with other products to form a complete solution. This presentation provides a high level discussion on developing and invoking web services which are compatible with Master Data Management Collaboration Server, referred to as MDMCS throughout this presentation.



Terminology

- Aim
  - Developing web services
- Terminology
  - MDMCS
  - TOP
  - WSDL
  - WSDD
  - SOAP
  - Axis
  - RSA
- What is not covered
  - Comprehensive details on programming or implementation

2 Developing web services © 2013 IBM Corporation

The aim of this presentation is to familiarize you with the processes of enabling and developing web services which are compatible with MDMCS.

Through the course of this presentation, certain terminology is used that is unique to the product and web services, including TOP. TOP is an environment variable pointing to the installation directory of MDMCS.

WSDL is the Web Services Description Language. It is written in XML and is used to describe what a web service is designed to do.

WSDD is the Web Service Deployment Descriptor. It is an Axis specific web service deployment configuration file and can be used to specify resources that should be exposed as web services.

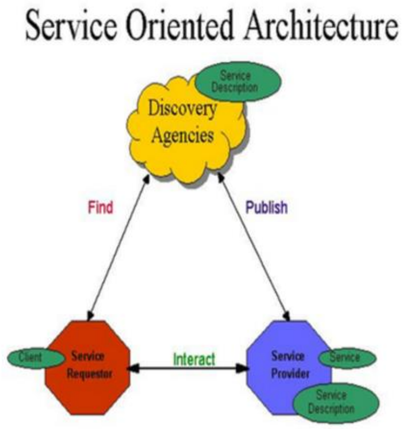
SOAP is an acronym for Simple Object Access Protocol. It is a transport protocol that sends XML messages using HTTP (which runs on top of TCP, typically on port 80).

Axis is an XML based, open source framework for developing web services. It is a global standard and currently MDMCS supports Axis 1.4.

RSA stands for Rational® Software Architect and is the IBM implementation of Eclipse platform to develop applications and web services.

This presentation is not meant to serve as a comprehensive programming or implementation guide to help you implement web services. Rather, it is meant to make you familiar with the whole process of implementing, deploying, and accessing web services.

Introduction



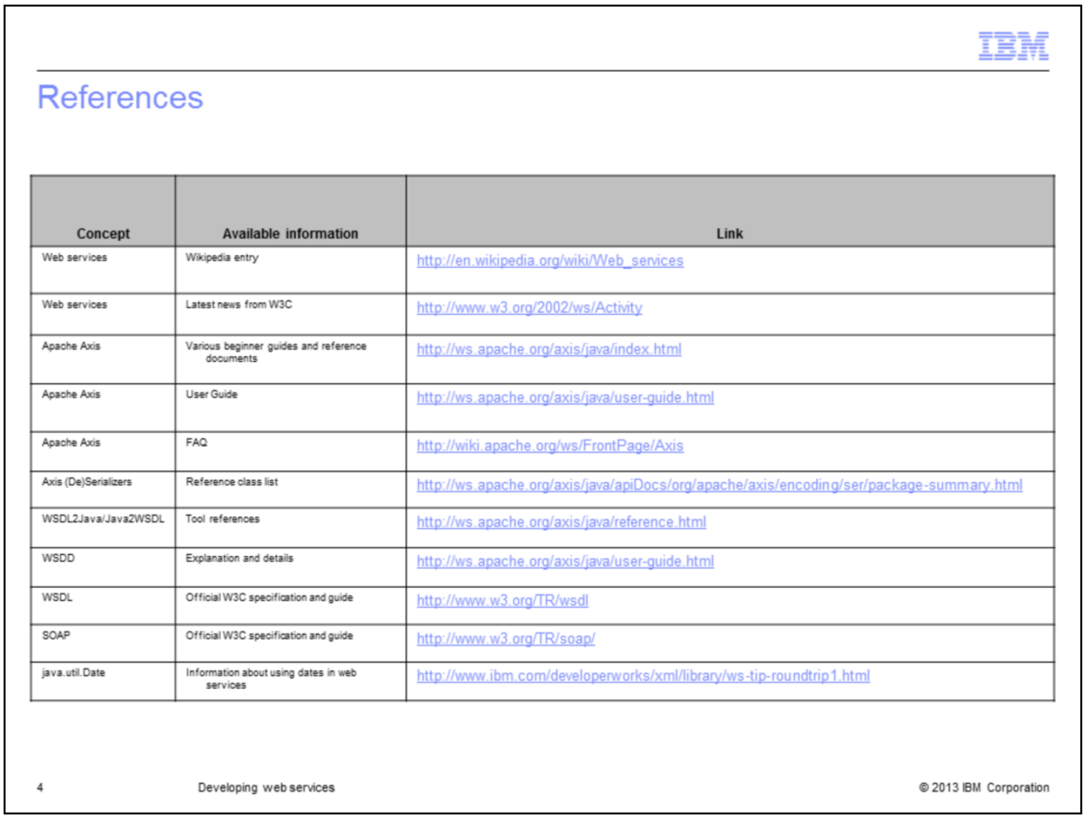
```
+--------------------------------------+
|         web service registry         |
|         (aka service broker)         |
|                (UDDI)                |
+--------------------------------------+
        ^                        ^
        |                        |
    (2) |                    (1) |
        |  (the client           |  (the web service
      WSDL  finds the          WSDL  provider publishes
        |   service              |   the web service)
        |   they want)           |
        |                        |
        v                        |
 +-----------+              +-----------+
 | service   |<---SOAP--->|  service  |
 | requestor |              |  provider |
 +-----------+      (3)     +-----------+
```

3 Developing web services © 2013 IBM Corporation

Web services are a way of communication between two electronic devices over the internet. There is often a business need to have two programs on one computer or to have different computers talk to each other. One is known as the server (waiting and listening for requests), and the other is the client (contacting the server when it needs something done that the server does). The client and the server can talk to each other in a variety of ways: sockets, pipes, text files and so on. The purpose of web services is to provide a generic, standardized, secure client and server paradigm for asking server programs on remote computers to perform some predefined action like computation, data retrieval, data storage and more.

See the diagram displayed on this slide for a high level overview on how web services work. At its simplest, a web service provider will create a web service and register itself with a web service registry. This is an optional step but if the provider does not register it, the independent requesters will not know how to invoke it. The second step is for the web service requester to find the web service they want along with the other essential details. This step is not needed if the requester already has the details of the web service to invoke. Finally, the client and provider will exchange information by way of SOAP messages.

To work with MDMCS, the client will already have the details of the provider and so only step three is needed.

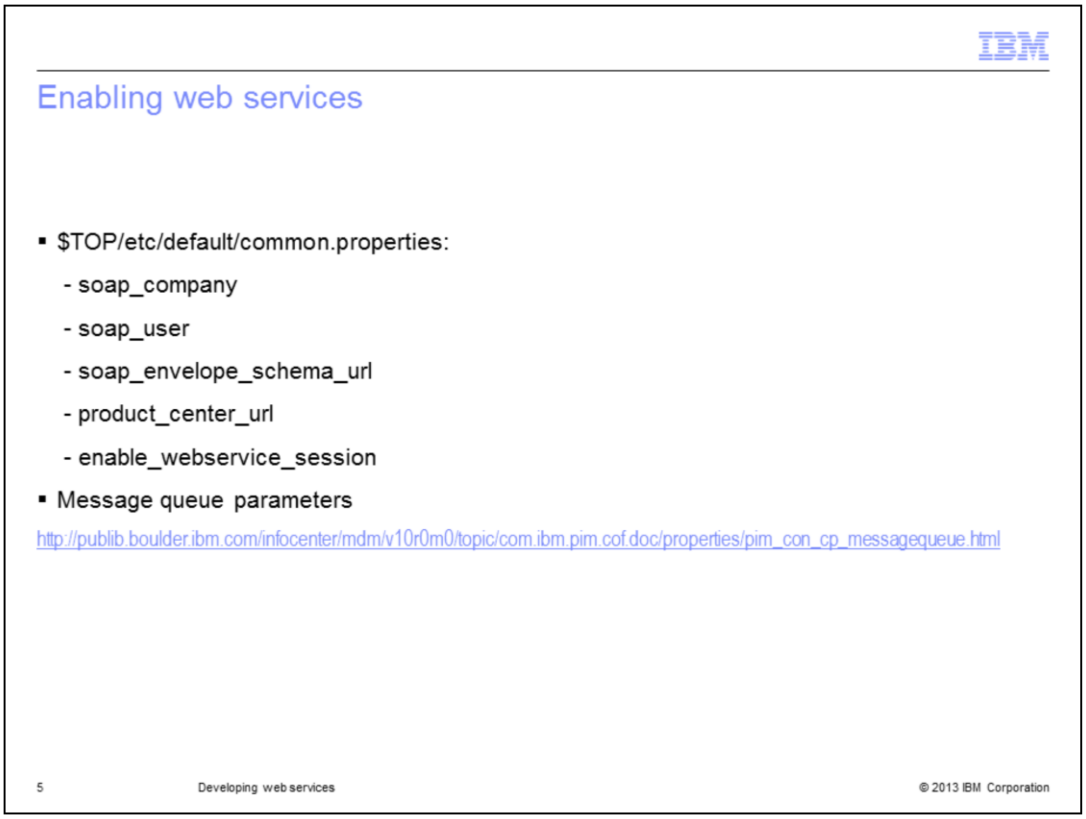


References

Concept	Available information	Link
Web services	Wikipedia entry	http://en.wikipedia.org/wiki/Web_services
Web services	Latest news from W3C	http://www.w3.org/2002/ws/Activity
Apache Axis	Various beginner guides and reference documents	http://ws.apache.org/axis/java/index.html
Apache Axis	User Guide	http://ws.apache.org/axis/java/user-guide.html
Apache Axis	FAQ	http://wiki.apache.org/ws/FrontPage/Axis
Axis (De)Serializers	Reference class list	http://ws.apache.org/axis/java/apiDocs/org/apache/axis/encoding/ser/package-summary.html
WSDL2Java/Java2WSDL	Tool references	http://ws.apache.org/axis/java/reference.html
WSDD	Explanation and details	http://ws.apache.org/axis/java/user-guide.html
WSDL	Official W3C specification and guide	http://www.w3.org/TR/wsdl
SOAP	Official W3C specification and guide	http://www.w3.org/TR/soap/
java.util.Date	Information about using dates in web services	http://www.ibm.com/developerworks/xml/library/ws-tip-roundtrip1.html

4 Developing web services © 2013 IBM Corporation

Further details of web services are beyond the scope of this presentation. Review any of the links displayed on this slide for such details.



Enabling web services

- $TOP/etc/default/common.properties:
  - soap_company
  - soap_user
  - soap_envelope_schema_url
  - product_center_url
  - enable_webservice_session
- Message queue parameters

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.cof.doc/properties/pim_con_cp_messagequeue.html

5 Developing web services © 2013 IBM Corporation

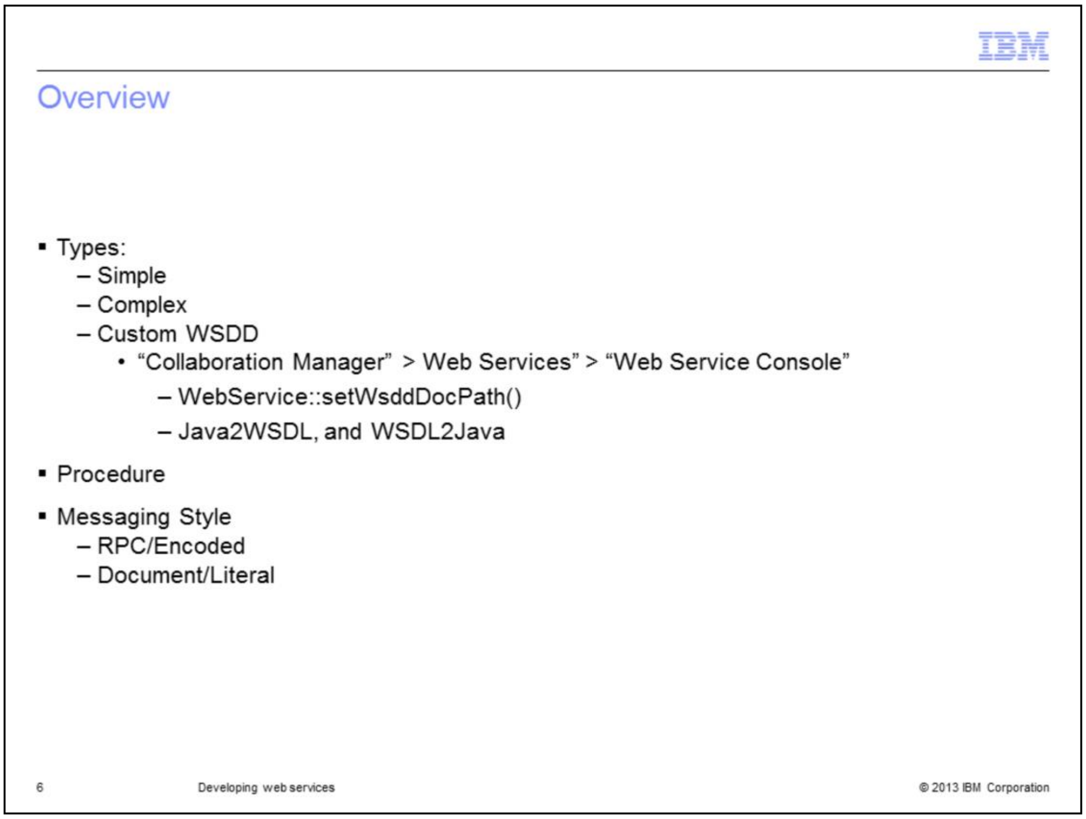
To enable web services, you have to modify these parameters in the common.properties file. The soap_company parameter defines the company credentials, and soap_user parameter defines the user credentials. When SOAP services are run, they use this company and user to access the database and run scripts.

The soap_envelope_schema_url parameter defines the URL of schema for the SOAP envelope and the port number.

The product_center_url parameter defines the fully-qualified URL, including the port number of the website where you should point your browsers to to access your MDMCS instance.

The enable_webservice_session parameter enables the web service session.

There are some message queue parameters that define how your system handles inbound and outbound messaging with external sources or destinations including web servers. Review the link displayed on the slide for more details.


Overview

- Types:
  - Simple
  - Complex
  - Custom WSDD
    - "Collaboration Manager" > Web Services" > "Web Service Console"
      - WebService::setWsddDocPath()
      - Java2WSDL, and WSDL2Java
- Procedure
- Messaging Style
  - RPC/Encoded
  - Document/Literal

6 Developing web services © 2013 IBM Corporation

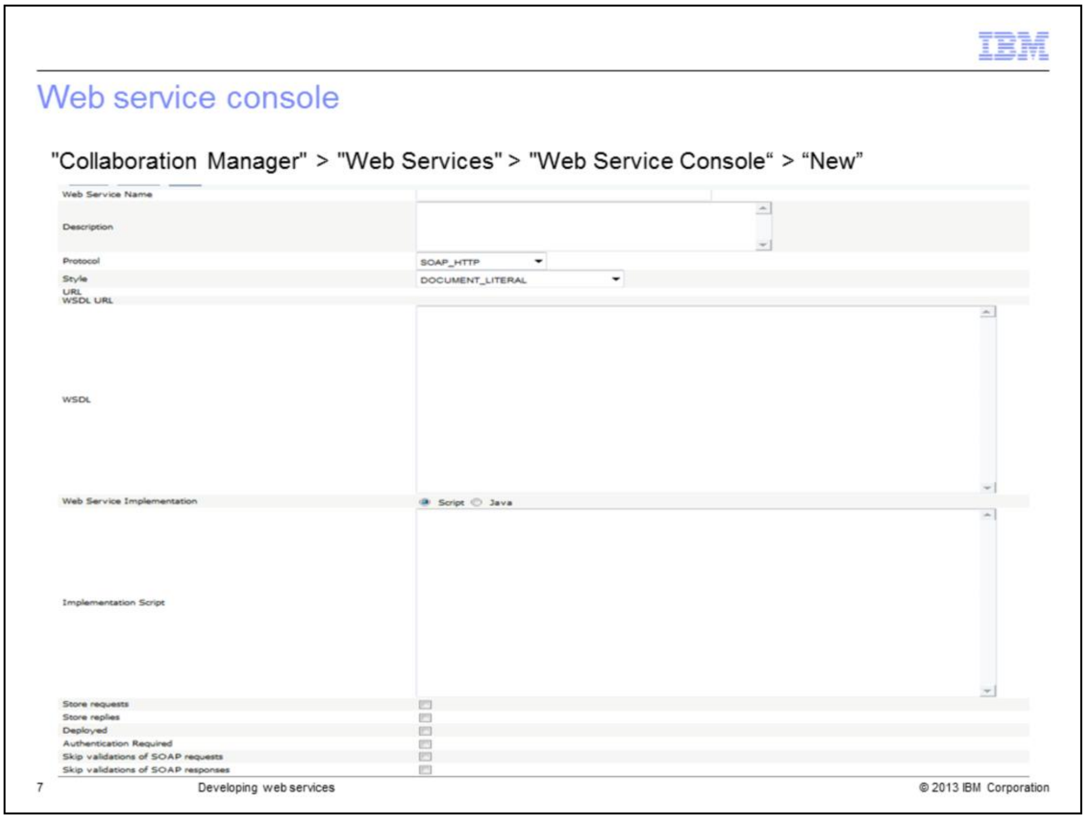
Web services can be broadly categorized into two types - simple and complex. These types are based on the type of data that is being exchanged.

Simple web services are ones where only simple types, such as string and int, are sent and received as arguments and returns from methods. MDMCS handles the generation of WSDL and WSDD for the deployment of the service.

Complex web services are used to exchange information using complex data object types when simple ones are not sufficient. To deploy complex web services, the web services author needs to provide a custom WSDD file which is specified in either the WSDD section of web service console or set programmatically by way of Java APIs. To author your own WSDD, you must have a good understanding of web services, the Java2WSDL, and WSDL2Java tools.

Leveraging web services is a three step process. First you have to write code, create WSDL, and WSDD files to implement it. Then you have to deploy it and finally access it. All these are discussed in greater detail on the next slide.

The messaging styles supported by MDMCS are document/literal or rpc/encoded. This dictates how to translate a WSDL binding to a SOAP message. The details of these styles are beyond the scope of this presentation but the author of the web service needs to decide which one to adopt. Rpc/encoded is simple but has relatively poor performance and is not WS-I compliant. Document/literal is more secure, is WS-I compliant but is more complicated.



Web service console

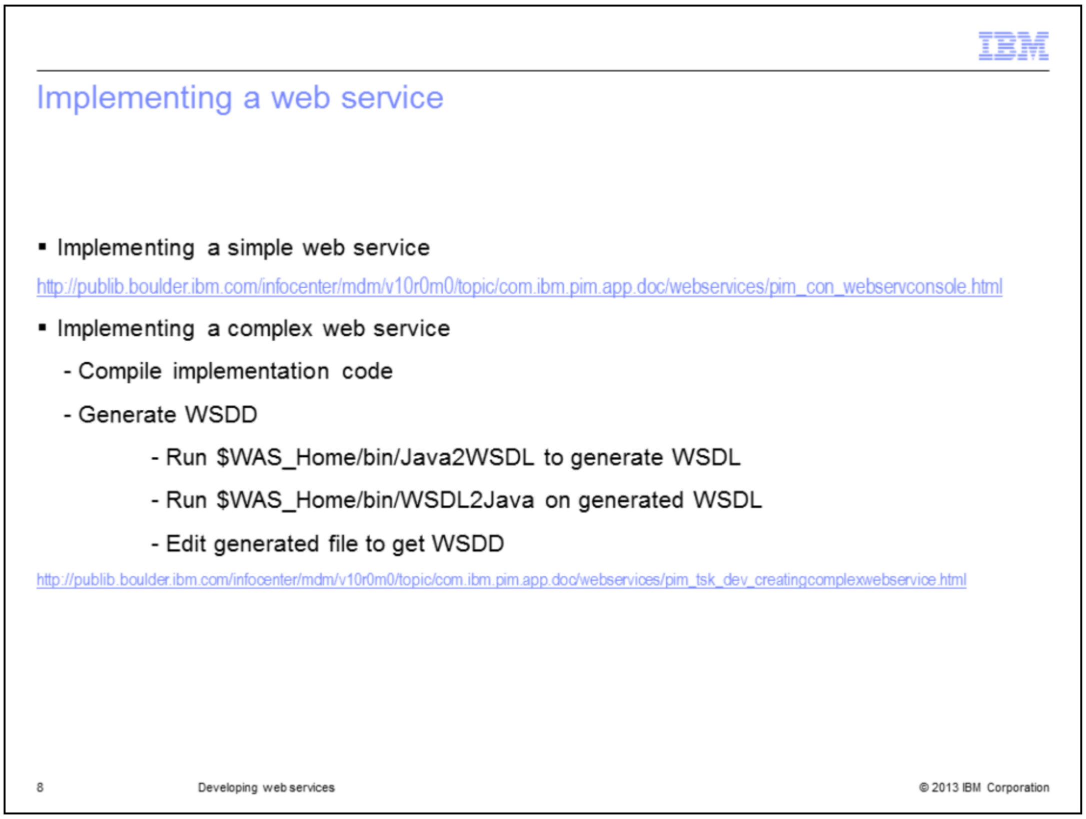
"Collaboration Manager" > "Web Services" > "Web Service Console" > "New"

Web Service Name	
Description	
Protocol	SOAP_HTTP
Style	DOCUMENT_LITERAL
URL	
WSDL URL	
WSDL	
Web Service Implementation	[✓ Script] Java
Implementation Script	
Store requests	☐
Store replies	☐
Deployed	☐
Authentication Required	☐
Skip validations of SOAP requests	☐
Skip validations of SOAP responses	☐

7 Developing web services © 2013 IBM Corporation

The web services console provides the administrator the ability to create and manage web services that are exposed by MDMCS. The MDMCS team encourages using the console to manage your web services for security, simplification, and to reduce the required maintenance. The web service console displays the web services as rows in a table. You can also define new web services or modify existing ones. This slide displays a screen capture of the web service definition page.

Most of the prompts are self explanatory. Some of the important features of this screen are protocol which is always SOAP_HTTP. Style, which can be document_literal or rpc_encoded. The web services description file contains the WSDL file and is published to the default HTTP server. The web services implementation script is the script invoked by an incoming web service request. This script decodes the incoming request parameters, executes a query against the data model to fetch the appropriate information, and formulates a response message.



Implementing a web service

- Implementing a simple web service

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_con_webservconsole.html

- Implementing a complex web service
  - Compile implementation code
  - Generate WSDD
    - Run $WAS_Home/bin/Java2WSDL to generate WSDL
    - Run $WAS_Home/bin/WSDL2Java on generated WSDL
    - Edit generated file to get WSDD

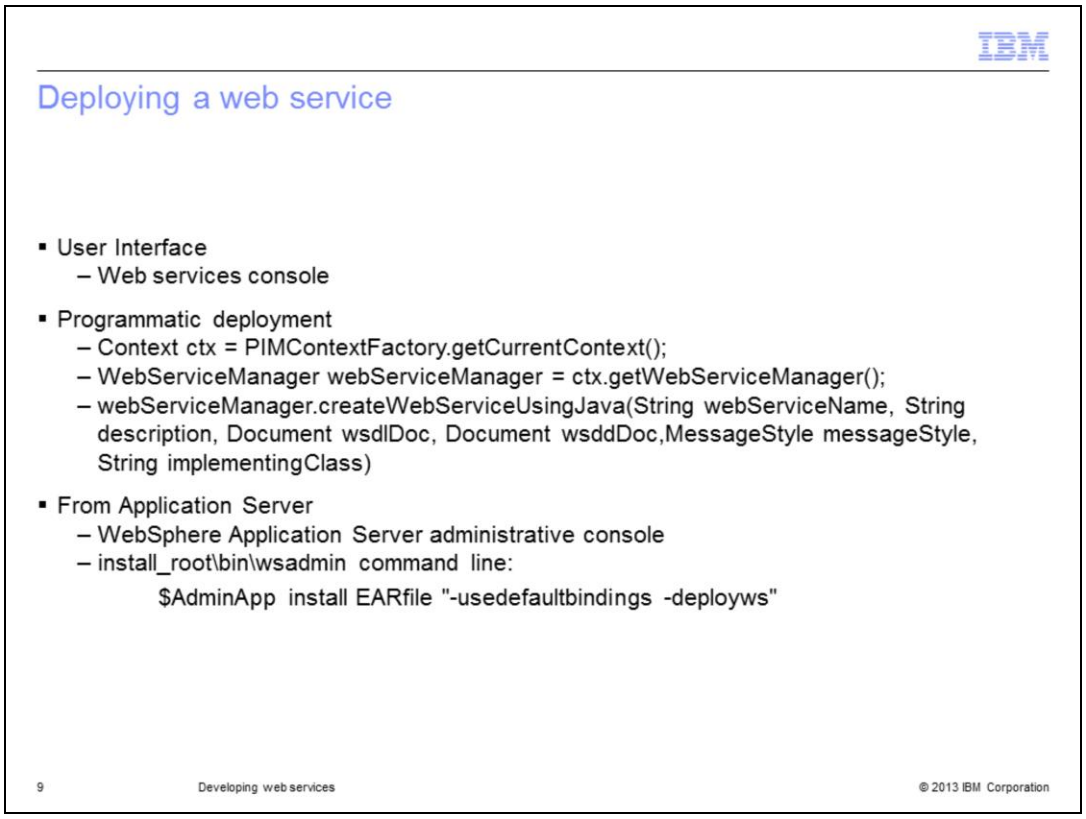
http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_dev_creatingcomplexwebservice.html

8 Developing web services © 2013 IBM Corporation

Simple web services are ones that use only simple types such as string and int. You can deploy a simple web service on MDMCS without any customization of WSDD or WSDL. All that is required is to go to the web service definition page and populate it with the implementation code with either Java or a native application script. Refer to the Information Center link displayed on this slide for a sample implementation script and WSDL document.

Complex web services are used to use custom data types to set and retrieve data which cannot be classified into simple ones like string and int. Firstly, to create a complex web service, write and compile the code for creating a complex web service. Since by definition, such web services do not use standard data types, the product cannot auto generate the WSDL and WSDD file needed for deployment. Therefore, to generate the WSDL file, run Java2WSDL tool which is provided by Apache Axis foundation. Then use WSDL2Java tool, also provided by Apache Axis, on the generated file. Finally, edit the file as needed to get your WSDD file. Now your web service is ready for deployment in the web service console.

Note that these two Java2WSDL and WSDL2Java tools are bundled with most application servers. For WebSphere® Application Server, these can be found under the bin directory of the installation.


Deploying a web service

- User Interface
  - Web services console
- Programmatic deployment
  - Context ctx = PIMContextFactory.getCurrentContext();
  - WebServiceManager webServiceManager = ctx.getWebServiceManager();
  - webServiceManager.createWebServiceUsingJava(String webServiceName, String description, Document wsdlDoc, Document wsddDoc,MessageStyle messageStyle, String implementingClass)
- From Application Server
  - WebSphere Application Server administrative console
  - install_root\bin\wsadmin command line:

```
$AdminApp install EARfile "-usedefaultbindings -deployws"
```

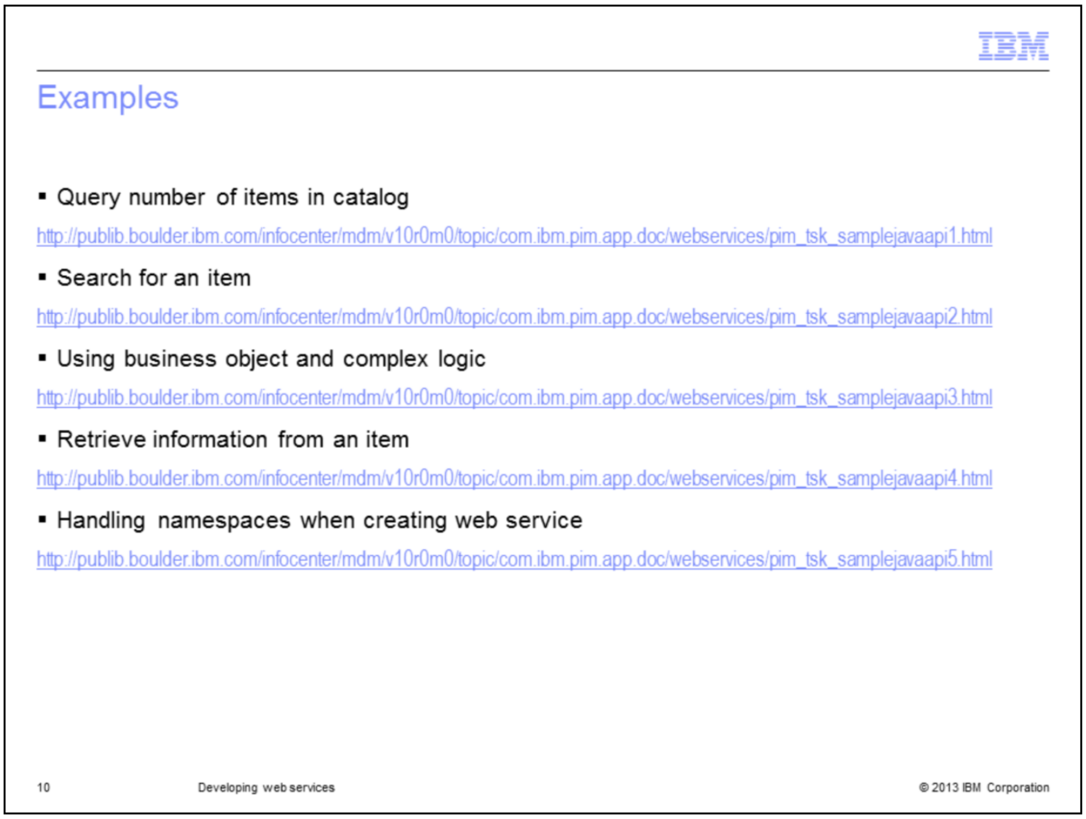
9 Developing web services © 2013 IBM Corporation

After implementation, you must deploy a web service to be able to use it. You have three options for deploying a web service; from the user interface, to do so programmatically, or from the application server.

To deploy a web service from the user interface, go to the new web service creation screen in the web services console. Fill out all the required prompts and save it. Your web service is deployed.

To deploy programmatically, you can either use the createWebService script operation or createWebServiceUsingJava JAVA API to deploy these web services. A sample JAVA API code to deploy the web service is displayed on this slide.

Finally, you can create a .EAR file of the web services you have and then deploy them together from the application server. For WebSphere Application Server, you can do so from either the administrative console GUI or the wsadmin command prompt. The commands to use are displayed on this slide. This process is discussed in more detail on slide 11.



Examples

- Query number of items in catalog

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_samplejavaapi1.html

- Search for an item

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_samplejavaapi2.html

- Using business object and complex logic

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_samplejavaapi3.html

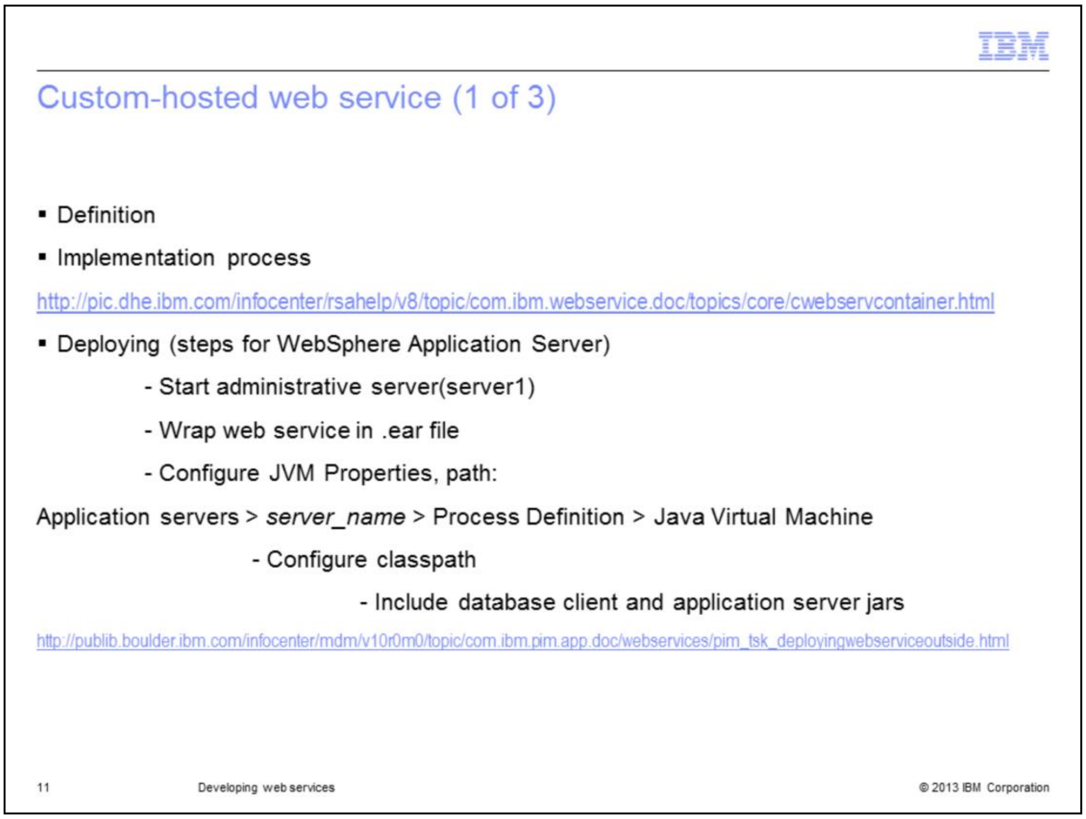
- Retrieve information from an item

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_samplejavaapi4.html

- Handling namespaces when creating web service

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_samplejavaapi5.html

The Information Center contains very comprehensive examples on how to implement and deploy simple and complex web services. Review these samples and use them as a template to build your own web services.


Custom-hosted web service (1 of 3)

- Definition
- Implementation process

http://pic.dhe.ibm.com/infocenter/rsahelp/v8/topic/com.ibm.webservice.doc/topics/core/cwebservcontainer.html

- Deploying (steps for WebSphere Application Server)

  - Start administrative server(server1)
  - Wrap web service in .ear file
  - Configure JVM Properties, path:

Application servers > *server_name* > Process Definition > Java Virtual Machine

- Configure classpath
  - Include database client and application server jars

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_tsk_deployingwebserviceoutside.html

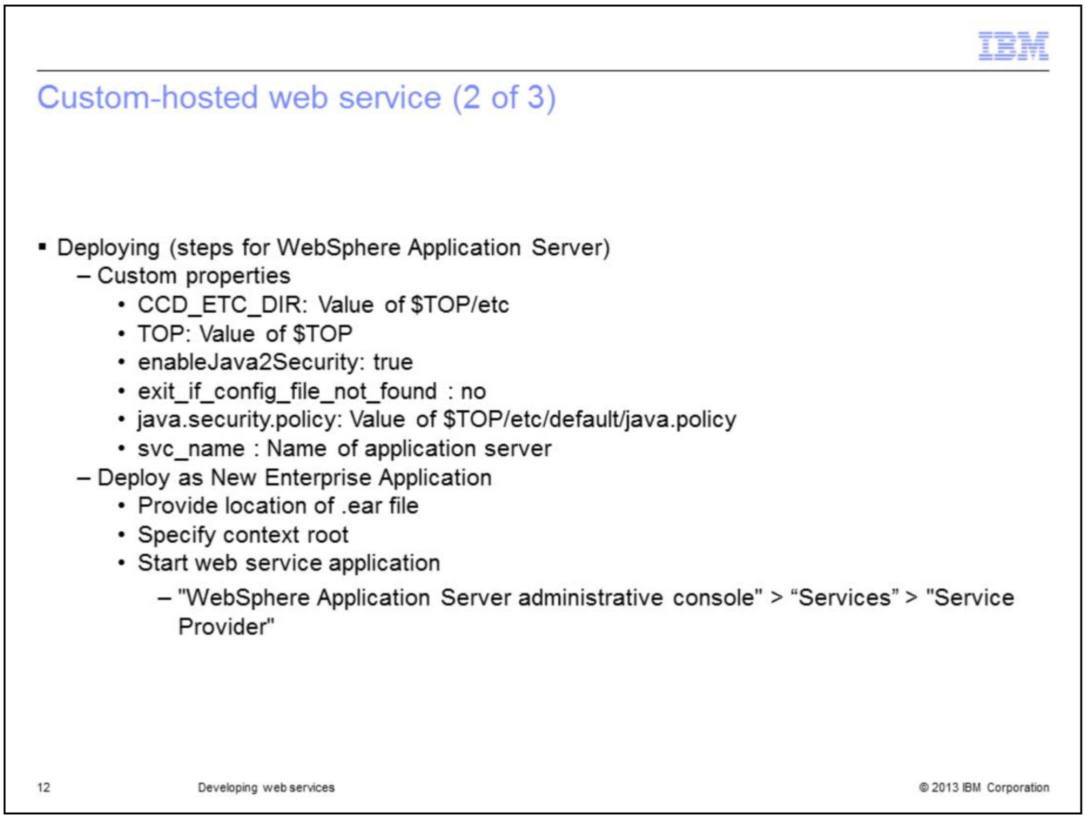
11 Developing web services © 2013 IBM Corporation

A service that is hosted on a server other than MDMCS is known as a custom-hosted web service. MDMCS supports such web services. Since it is hosted outside MDMCS, it can be implemented using standard web service development processes. Refer to the Information Center link for Rational Software Architect which describes the process of developing a web service.

All custom-hosted web services have to be manually deployed at the application server. This presentation only discusses the deployment process for WebSphere Application Server. Before starting, make sure that the administrative server, known as server1, is running. Then, wrap the web service or group of web services that you want to deploy into a .ear file. To ensure that the web service is deployed correctly, you need to configure certain JVM properties. These properties are located on the path displayed on this slide.

First, configure the classpath. Make sure that the tools.jar from the application server and the corresponding client database jars db2jcc.jar, db2jcc_license_cu.jar, ojdbc6.jar are part of the classpath. Along with these, make sure you add all the jars under $TOP/jars to the classpath.

Note that some of the jars under $TOP/jars are not needed. If you prefer to deploy only the ones which are necessary, you can get a list from the Information Center.



Custom-hosted web service (2 of 3)

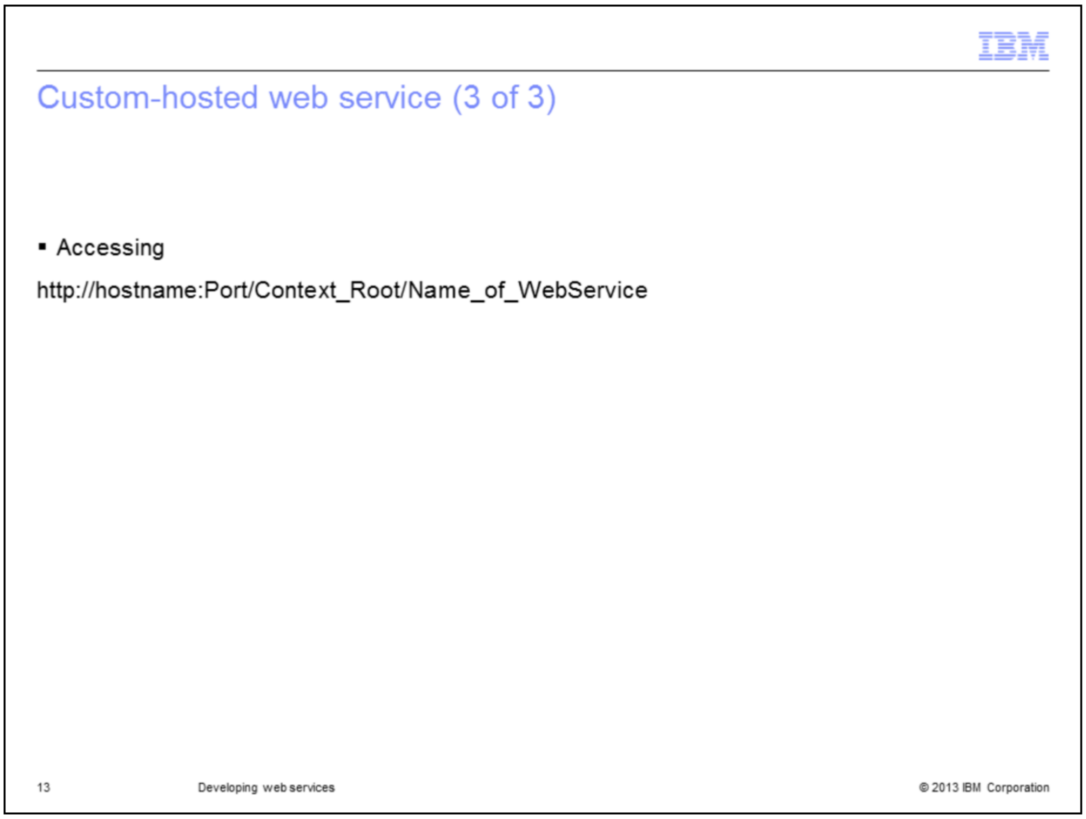
- Deploying (steps for WebSphere Application Server)
  - Custom properties
    - CCD_ETC_DIR: Value of $TOP/etc
    - TOP: Value of $TOP
    - enableJava2Security: true
    - exit_if_config_file_not_found : no
    - java.security.policy: Value of $TOP/etc/default/java.policy
    - svc_name : Name of application server
  - Deploy as New Enterprise Application
    - Provide location of .ear file
    - Specify context root
    - Start web service application
      - "WebSphere Application Server administrative console" > “Services” > "Service Provider"

12 Developing web services © 2013 IBM Corporation

The second step is to configure the custom properties in the JVM. Ensure that all the custom properties in your WebSphere Application Server administrative console are configured as displayed on this slide. Without these, the web service is not correctly deployed.

The final step is to deploy it as a “New Enterprise Application”. You can do so by providing the location of the .ear file, then specifying the context root and finally starting the web service application.

Your custom-hosted web service has been deployed.


Custom-hosted web service (3 of 3)

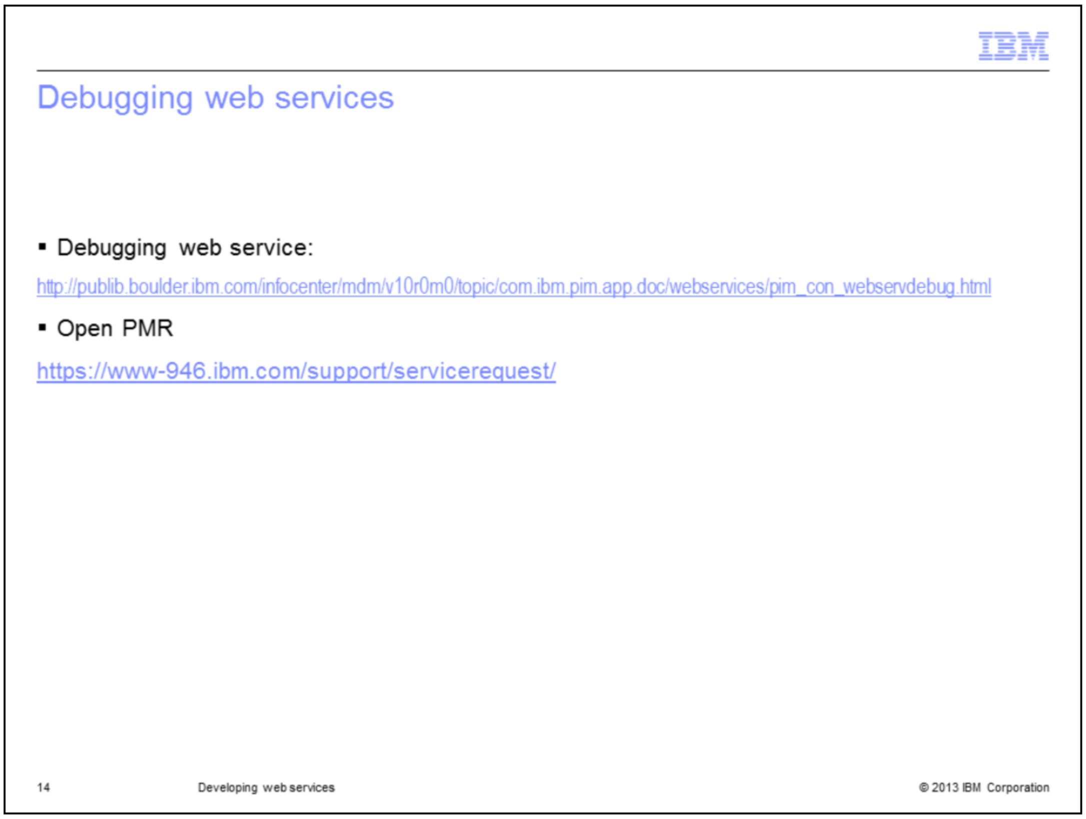
- Accessing

http://hostname:Port/Context_Root/Name_of_WebService

13 Developing web services © 2013 IBM Corporation

After you deploy the web service, you need to be able to access it to derive real use. You can do so by going to the URL of the type displayed on this slide. Replacing hostname, port, context root, and name of the web service with the actual ones will allow you to access it. Note that all the web services in the same .ear file will have the same context root, which is defined at the time of deployment.

It should be noted that you can restart the MDMCS Application Server multiple times without needing to redeploy the web services if you are using WebSphere Application Server. However, if you are using Weblogic, you need to redeploy it every time MDMCS Application Server service is restarted.



Debugging web services

- Debugging web service:

http://publib.boulder.ibm.com/infocenter/mdm/v10r0m0/topic/com.ibm.pim.app.doc/webservices/pim_con_webservdebug.html

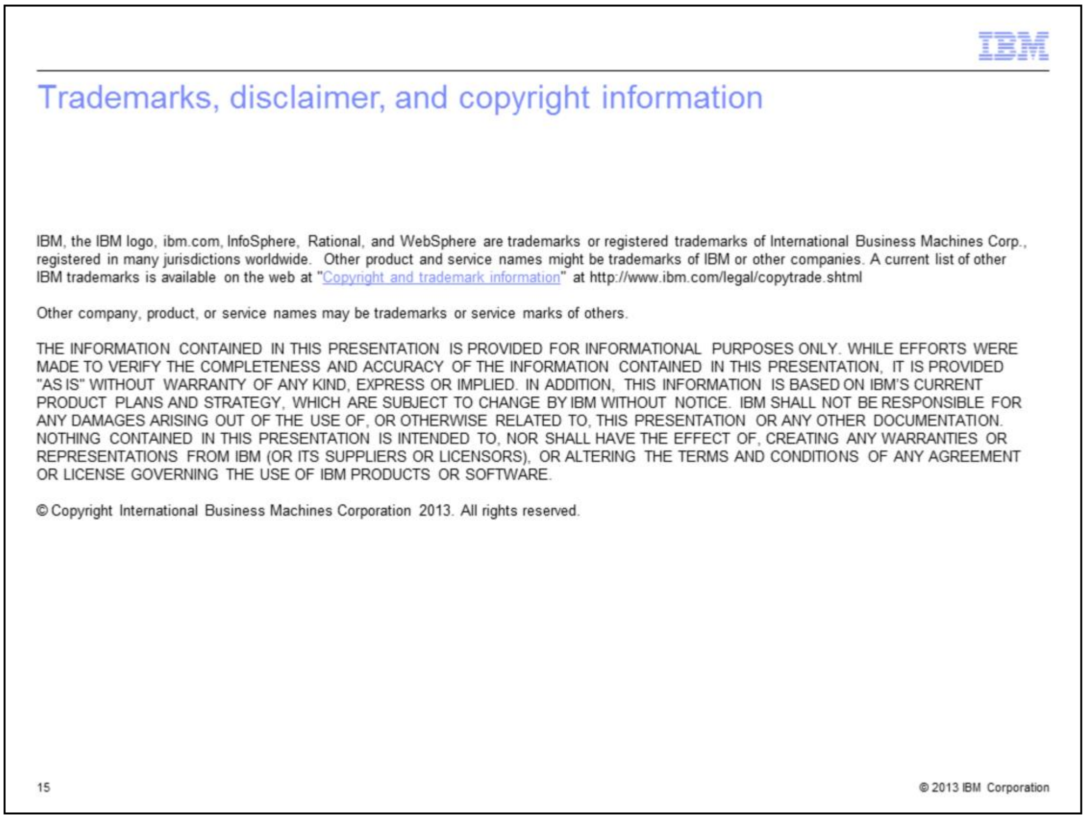
- Open PMR

https://www-946.ibm.com/support/servicerequest/

14 Developing web services © 2013 IBM Corporation

The Information Center also creates some very useful suggestions on how to debug a web service if you encounter any issues. These suggestion are valid for both simple and complex web services.

If you cannot narrow down the root cause using these suggestions, open a PMR and the IBM Support team is happy to assist you.



Trademarks, disclaimer, and copyright information