A Technical How-to Guide for Creating Components and Web Services in Rational Rapid Developer
June, 2003 Rev. 1.00

IBM® Rational® Rapid Developer Components & Web Services

Glenn A. Webster
Staff Technical Writer

Executive Summary

This document describes the IBM® Rational® Rapid Developer system for developing components and publishing and consuming Web services. The document does not presume any prior knowledge of Rational Rapid Developer.

A Rational Rapid Developer component is an encapsulated element of application functionality that is exposed through a well-defined interface. You can get data, set data, or perform actions by using the public interfaces as defined by the component developer or author. This enables the author to extend or alter the component's functionality and internals while at the same time provide the same interface contract to the component consumer (e.g., another application). Components also provide you with a means to publish Web services for use in other applications.

About IBM Rational

IBM Rational, formerly an independent company and now one of the IBM software brands, offers a comprehensive software development solution. The IBM Rational platform combines software engineering best practices, market-leading tools, and expert professional services, all of which drive rapid and continuous improvement in software development capability for on demand businesses.

In addition, IBM Rational offers more than 20 years experience in promoting and delivering integrated and open software systems, both of which are key characteristics of the on demand operating environment:

Integrated — IBM Rational has contributed considerable thought leadership and expertise in the areas of Service-Oriented Architecture (SOA), enterprise and software architecture, and heterogeneous platform support.

Open — IBM Rational has a long history in developing and supporting the goals of open computing. This includes development of the Unified Modeling Language (UML), now a standard for modeling applications, database design, and business processes. IBM Rational has promoted and participated in the development of a wide variety of open computing standards. It offers support for major programming languages and operating platforms, and it provides an extensive set of application programming interfaces for third-party tools interoperation.

Thousands of companies around the world have realized the benefits of the approach advocated by IBM Rational. Their processes are results-oriented, the artifacts they produce are well-designed and reusable, and they are working at higher levels of capability now required by the on demand era.

Contents

Component Modeling In Rational Rapid Developer

A Rational Rapid Developer component is an object that encapsulates application functionality, and is exposed through a well-defined interface. You can get data, set data, or perform actions by using the public interfaces defined by the component developer. This enables the author to extend or alter the component's functionality and internals while at the same time provide the same interface to the component consumer (e.g., another application). In the J2EE platform, a modeled component is constructed as an EJB Session Bean. This Session Bean can be called by any Java™ application.

Why Use Components?

Software re-use and modular development practices are the driving themes for using components. Two practical examples of components in action follow.

- Automating a business process — based on the transactions that have occurred during the business day, execute a specific business process at night. Using Rational Rapid Developer component modeling, you define an ObjectSpace transaction model with all the required business objects, attributes, and business methods for the process. You then mark the methods that serve as the callable interfaces, and then construct the component. In the J2EE platform, this constructed and deployed Session Bean can then be looked up using JNDI and invoked from any independent Java client application that is invoked by scheduling services such as cron.

- A business department needs to access current information from another department's running application. The access is API-based and not from a Web page. For example, a time tracking Web application that is used by department employees to track billable versus non-billable time. The accounting department, which sends invoices to the company's clients, also needs this billable information. In Rational Rapid Developer, the developers of the time tracking application could create a component and expose its interfaces so that an external application could invoke the component. Once the application is constructed and deployed, the accounting department application could then call the time tracking component to retrieve the information needed to process invoices.

Rational Rapid Developer is a development tool that allows developers to create applications and not worry about constantly changing technology

Specifying the Environment

Currently, component construction is only supported for the J2EE platform. Follow these steps to set up the environment.

1. In the Application Architect's Application Navigator, select the top level icon.
2. Select the icon for the active Partition Model.
3. Select the Technology Settings icon.
4. Select the Application Server tab.
5. Set the Application Server Technology to EJB With Servlets or EJB with Servlets With JSP.
6. Specify your EJB Server and its properties.
7. Specify your application server's deployment settings.

Creating a Component

Follow these steps to create a component.

1. In the Application Architect's Application Navigator, select the class with which the component is to be associated. We recommend that you associate the component with the business object that has primary participation in the transaction.
2. Select the Components tab.
3. Select the New button and name the new component (e.g., NumberComp).
4. Select the ObjectSpace button to create the transaction model for the new Component.
5. In the ObjectSpace, bring in all classes, attributes, and methods that your transaction requires and will utilize.
6. Right-click on the root class in the ObjectSpace and select "Add Local Method". A Local Method is a method associated to the ObjectSpace and can access anything modeled in the ObjectSpace. It is the required way for exposing component interfaces.
7. In the Local Method dialog, specify the Name, Definition, and write the method's code.
8. Select the OK button to close the dialog box.
9. Select the local method in the tree, then select the Interface checkbox in the Component section of the screen.
10. Select the OK button to close the ObjectSpace dialog box.
11. Press the Construct button to construct the component. You can now use this component in your application or any external application.

Using Components

To use a component you need to specify the physical location of the component's .jar file, which you will reference and add to the class path. This is done by:

- To make the component available to all partition models, add the component's .jar file to the Add to Class Path section of the Global Includes tab of the Application Package.

- To make the component available in a specific partition model only, select the partition model's name in the Application Navigator, then add component's .jar file to the Add to Class Path section of the Model Includes tab of the Technology Settings.

For example:

e:\myapp\EJBCOMP\build\weblogic5\NumberComp\NumberComp.jar

Alternatively, you can specify the package in which the component resides to make the classes easier to reference in your code. You do this by adding the package name to the Import Java Packages section of the Global Includes tab. For Rational Rapid Developer constructed components the package name follows the naming convention below.

"com.myapp.cmp.componentname.*"

where;

com = domain name type (Rational Rapid Developer defaults to this value)

myapp = the name of the Rational Rapid Developer application

cmp = stands for component (Rational Rapid Developer defaults to this value)

componentname = the name of the component in Rational Rapid Developer.

Example: Using a Component in the Same Application

The following is a simple example that shows how to use a custom method to access a component.

```
<%
java.util.Properties props=System.getProperties();
javax.naming.Context nCtx=new
javax.naming.InitialContext(props);
com.componentapp.cmp.NumberComp.NumberCompHome home=
      (com.componentapp.cmp.NumberComp.NumberCompHome)
       nCtx.lookup("ComponentApp.NumberCompHome");
com.componentapp.cmp.NumberComp.NumberComp
objComp=home.create();
String strCompReturn=objComp.returnStr();
response.Write(strCompReturn);
%>
```

where;

- com.componentapp.cmp.NumberComp = the package name.
- com.componentapp.cmp.NumberComp.NumberCompHome = the home interface.
- com.componentapp.cmp.NumberComp.NumberComp = the remote interface.
- returnStr() = the interface to the component (local method defined while creating the component).

NOTE: The code example above uses explicit class referencing for the component.

Example: Using a Component From an External Application

The following is a java program named testCmp.java, used to demonstrate a component being accessed from a java standalone program.

```
import javax.ejb.*;
import java.rmi.RemoteException;
import java.rmi.Remote;

public class testCmp
{
  public static void main(String args[])
  {
       try
         {
          testCmp objtestCmp=new testCmp();
          objtestCmp.callMethod();
         }
       catch(Exception e)
```

```
{
       System.out.println("Cannot call component");
}
}

public void callMethod() throws Exception
{
       java.util.Properties props=System.getProperties();


       //Properties of remote m/c, i.e., one having the
component running
       // in this case, it is the local machine
       String url="t3://localhost:7001";

   //Note this is a WebLogic specific example.
       //Weblogic username and password, set in the
Weblogic properties file
       String user="system";
       String password="system";
       try
       {
      System.out.println("In the function callMethod()");

props.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
                "weblogic.jndi.WLInitialContextFactory");

       props.put(javax.naming.Context.PROVIDER_URL,url);

       props.put(javax.naming.Context.SECURITY_PRINCIPAL,
user);

props.put(javax.naming.Context.SECURITY_CREDENTIALS,passw
ord);
   javax.naming.Context nCtx1 = new
javax.naming.InitialContext(props);

      try
          {
              System.out.println("Looking up home
interface
                 for NumberCmpHome bean..");


       com.componentapp.cmp.NumberComp.NumberCompHome
home=
       (com.componentapp.cmp.NumberComp.NumberCompHome)
        nCtx1.lookup("ComponentApp.NumberCompHome");
              com.componentapp.cmp.NumberComp.NumberComp
objComp=home.create();
              System.out.println("\n Calling the method
in the component...");
              String strCompReturn=objComp.returnStr();
              System.out.println(strCompReturn);
```

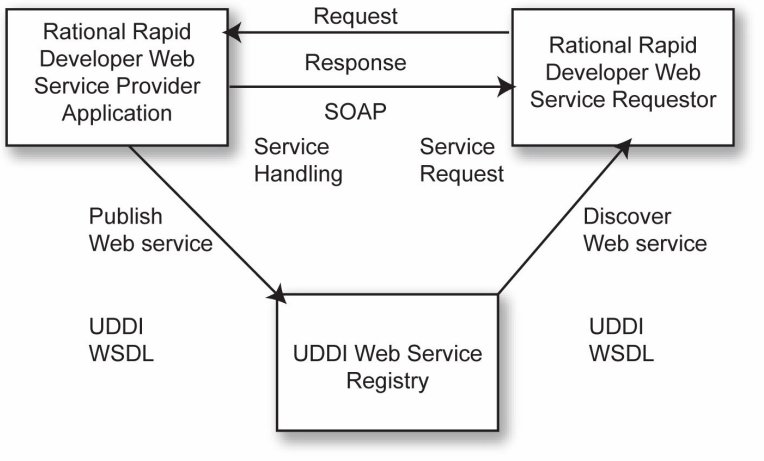
```
        }
           catch(Exception e)
        {

              System.out.println("Cannot find home
interface");
           }
        }
        catch(Exception e)
        {
           System.out.println(e.getMessage()+"\n");
           System.out.println("Cannot create
context"+e.toString());
        }

  }
}
```

Note: The code example above uses explicit class referencing for the component.

Web Services in Rational Rapid Developer

Web services are a new form of Web application. They are self-contained, self-describing applications that can be published, discovered, and consumed on the Internet. A Web service can perform any function, ranging from a simple inquiry request to a complex business process. Once a Web service is published, other applications or even other Web services can discover and consume the Web service. Rational Rapid Developer enables the developer to leverage this enterprise capability by creating, publishing, discovering, and consuming Web services. This leading-edge capability insulates the developer from the underlying technology and vendor issues. This capability will evolve as Web services standards emerge and mature.

Rational Rapid Developer can construct applications to be Web service providers or Web service consumers. Rational Rapid Developer can also discover and publish Web services to a UDDI registry. The following diagram illustrates these capabilities and mechanisms utilized.



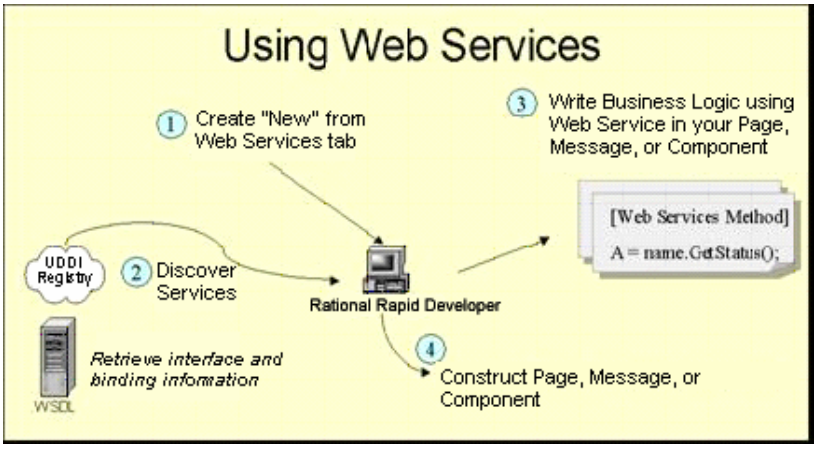
To use a Web service, you need to follow some very simple steps.

1. Create a new Web service name that you will reference in your application.
2. Discover a Web service from a UDDI registry of choice or from any WSDL location.
3. From your business logic, call this newly discovered Web service just as you would call any business object method.
4. Construct and test. Rational Rapid Developer isolates the developer from UDDI API calls, WSDL parsing, binding, SOAP invocation/response details, and vendor specific nuances.

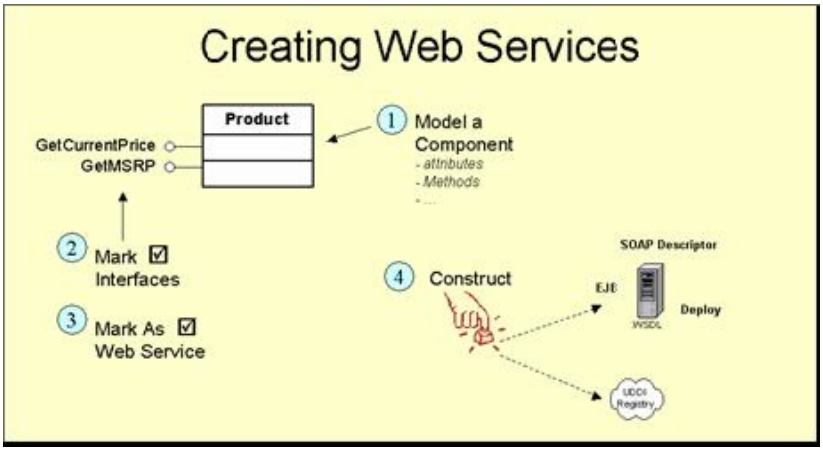
Notes:

- Web services are available using J2EE application servers only.
- Web services access (calling a Web service through either UDDI discovery of business names, or loading a Web service WSDL) is available using all of the Rational Rapid Developer supported J2EE application servers.

Rational Rapid Developer supports Web services publishing only under IBM WebSphere Application Server, BEA WebLogic and Oracle9iAS (version 9).



To create a Web service that is utilized by other applications, you first model a component, then select the public interfaces that you want to expose as Web service methods. Rational Rapid Developer constructs the appropriate session EJBs, classes, SOAP descriptors, and WSDL, and deploys for you automatically. Rational Rapid Developer optionally registers the Web service to a UDDI registry.



UDDI Registry

The Universal Description, Discovery, and Integration (UDDI) is an XML-based registry for businesses worldwide to list their Web services on the Internet. UDDI streamlines online transactions by enabling companies to find each other on the Web and make their systems interoperable for e-commerce.

You can discover Web services and register Web services with the IBM UDDI registry and the Microsoft UDDI registry. You also can publish to any UDDI standard public or private registry. To publish to these UDDI

registries, you need to establish an account (user id and password) with the registry service provider.

Configuring UDDI

1. In the Application Navigator, click on the plus (+) sign next to Partition models:
2. Select the plus (+) sign next to the name of the partition model for which you want to specify technology settings:
3. Click on Technology Settings. Rational Rapid Developer displays the Technology Settings page.
4. Select the UDDI tab.
5. Select a UDDI address from the UDDI Address list in the Discovering area of the page to utilize existing Web services in your application. Rational Rapid Developer includes a list of predefined Web service publishers. To use a UDDI address that is not in the list, type the UDDI address in the UDDI Address field.
6. If you want to publish a Web service from your application, you need to specify the Publishing properties.
    a. Type in the UDDI Secure Address field the secure address of the UDDI business registry to which you want to publish the Web service.
    b. Type in the User Name field the user name for a user registered with the business registry specified in the UDDI Secure Address field.
    c. Type in the Password field the password for the user specified in the User Name field.
    d. Type in the Server Name field the URL of the server that will publish the Web service. This is the URL of the server that is to be accessed at runtime by Web service requesting applications, and to which the WSDL file will be published.

Discovering Web Services

You can discover Web services in several ways:

- From a UDDI registry
- By loading a Web services WSDL file from a URL
- By browsing to a WSDL file on a network. Web Services Description Language (WSDL) files are XML-based files that are used to describe the services a business offers, and provide detailed binding information.

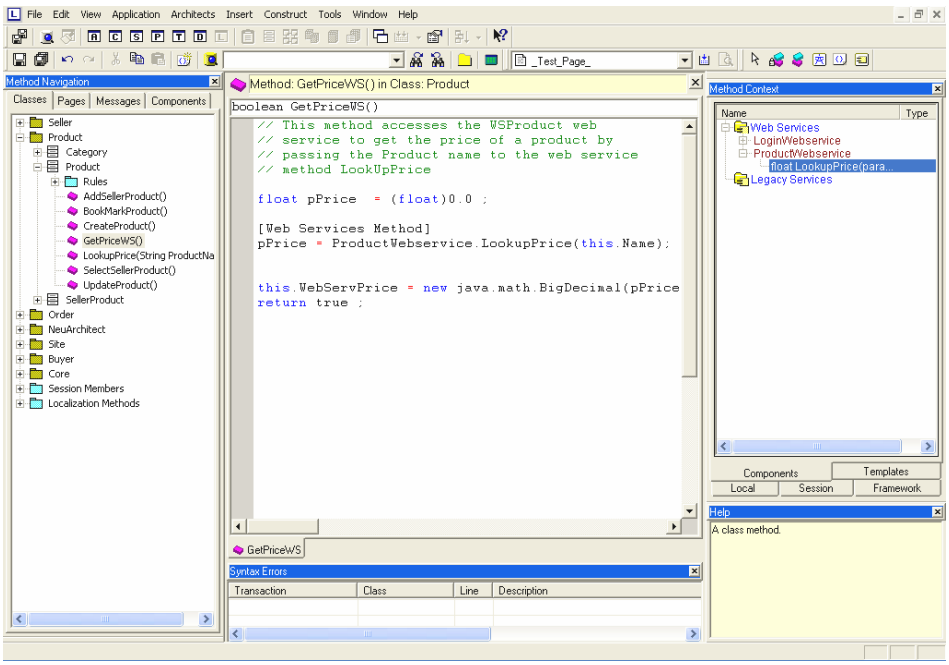
To discover the WSDL Location of a UDDI-registered Web Service:

1. In the Application Navigator, select the Application Package.
2. Select the Web Services tab.
3. Select the New button to add a new Web service.
4. Name the Web service.
5. Select the Discover button. Rational Rapid Developer displays the Discover Web Service dialog box.
6. Type the name of the business that publishes the Web service in the Business Name field.
7. Select the Discover button. Rational Rapid Developer displays the Web services discovered for the business that you specified.
8. In the Services list, select a method name, then select the OK button.

Using Web Services

To use the discovered Web services in a method:

1. In the Application Navigator, select the class that contains the method in which you want to use a Web service.
2. Select the Methods tab.
3. Select the method name in the Name list.
4. Select the Logic Architect button.
5. Expand the Web Services tree on the right side of the Logic Architect screen and drill down to the methods that you want to use.
6. Double-click on the method. Rational Rapid Developer places the Web Service call into the body of your custom method. Rational Rapid Developer places a bracketed “[Web Services Method]” tag directly before the Web services call. This is required so that Rational Rapid Developer can generate the appropriate SOAP API calls at construction time.



Change the input and return parameters to suit your business need.

Publishing Web Services

1. In the Application Architect's Application Navigator, select the class in which you want to define the component that will provide the Web service.
2. Create a component.
3. In the Select Construction pattern portion of the page, select a construction pattern that supports Web services.
4. Select the ObjectSpace button on the Component tab.
5. Add an ObjectSpace local method, and specify the code for the Web service.
6. Select the Interface and Web Service check boxes:
7. Select the OK button to close the ObjectSpace dialog box.
8. In the Application Architect's Application Navigator, click on the + sign to the left of the active partition model.
9. Select Technology Settings.
10. Select the UDDI tab.
11. In the Web Services portion of the page, select the component in the Name list.
12. In the UDDI Secure Address field, select the UDDI address to which you would like to publish the Web service.
13. Select the Register to UDDI button. Rational Rapid Developer registers the Web service, and automatically fills in the values for the UDDI Service Key, UDDI tModel key, and Binding key fields.

Conclusion

Rational Rapid Developer provides a visual development environment for modeling services that can be constructed as components. Components have user-defined interfaces and are automatically constructed and deployed. Components can be invoked by any J2EE application, and are ideal for reuse across the enterprise.

The Rational Rapid Developer integrated visual modeling and automated construction system enables organizations to rapidly create, publish, discover and integrate Web services. Modeled components can be exposed as Web services, which can be automatically constructed and registered to a UDDI registry of choice. This capability also allows you to easily discover and utilize any published Web services in your Rational Rapid Developer application.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the United States of America
01-03 All Rights Reserved. Made in the U.S.A.

The IBM home page on the Internet can be found at **ibm.com**

IBM software integrated solutions

IBM Rational supports a wealth of other offerings from IBM software. IBM software solutions can give you the power to achieve your priority business and IT goals.

- DB2® software helps you leverage information with solutions for data enablement, data management, and data distribution.
- Lotus® software helps your staff be productive with solutions for authoring, managing, communicating, and sharing knowledge.
- Tivoli® software helps you manage the technology that runs your e-business infrastructure.
- WebSphere® software helps you extend your existing business-critical processes to the Web.
- Rational® software helps you improve your software development capability with tools, services, and best practices.

Rational software from IBM

Rational software from IBM helps organizations create business value by improving their software development capability. The Rational software development platform integrates software engineering best practices, tools, and services. With it, organizations thrive in an on demand world by being more responsive, resilient, and focused. Rational's standards-based, cross-platform solution helps software development teams create and extend business applications, embedded systems and software products. Ninety-eight of the Fortune 100 rely on Rational tools to build better software, faster. Additional information is available at www.rational.com and www.therationaledge.com, the monthly e-zine for the Rational community.

Part No. TP032