A Technical Discussion of Legacy Integration
10/02/03

IBM® Rational® Rapid Developer A Guide to Legacy Integration

Version 2

By Jeff Douglas
Advisory Software Engineer
jeffdouglas@us.ibm.com

Table of Contents

Introduction

This guide presents the legacy integration capabilities of IBM® Rational® Rapid Developer. It includes:

- **A limited education of legacy integration and secondly,**
- **Integration techniques available for Rational Rapid Developer customers, and how to select the right one(s) for use.**

What Do We Mean by Legacy Integration?

Legacy integration refers to the methods and transportation techniques used to access, and perhaps update, pre-existing data and applications from an existing computer system, often an IBM mainframe. Integration usually involves databases but it is not limited to such, as it may be desirable to integrate messaging and existing application programs, as well.

In the context of application development, legacy integration often means the creation of new front-ends, business logic, etc., that need to be delivered in a Web-based application that uses significant elements of already existing legacy applications. This is commonly referred to as a "composite" application. These composite applications may require data from a mainframe database such as IMS or VSAM, transaction logic from a CICS program or IMS transaction, messages from IBM WebSphere® MQ or a JMS message transport, other elements from an SAP or other ERP application, and finally, a new user interface for viewing in an HTML or WML client. Today, these new composite applications are generally created for either J2EE or .NET technology.

What are the Benefits of Legacy Integration?

The most significant benefit is continued use of the existing investments in generally smooth-running systems. Other benefits include retaining existing human legacy programming talent, minimal retraining of end-users, continued use of existing mainframe, network, and infrastructure assets. In essence, legacy integration provides minimal disruption and cost, and maximum reuse.

What Others are Saying

- "The cheapest application is the application you don't have to write," observes Jim Sinur of the Gartner Group. "One way to unshackle from legacy applications is to rebuild applications on a new platform, with a different language, different communications, and different connectivity protocols. The problem is that, unless the application is small, and resources are plentiful, the time and expense required in rebuilding from scratch is likely to be prohibitive." [1]

Thus the benefits of extending conventional host-based applications from proprietary desktops and networks include:

- Potential for financial savings by cutting network and terminal costs.
- Bonus return on the original investment in legacy systems.
- Opportunities for electronic commerce using legacy databases. Through "Weblications," businesses can create a presence on the Web and outsource functions like order taking and servicing to end clients.
- More satisfied users who get an attractive graphical interface.
- More integrated and efficient functionality.

We are Talking About More than Mainframes Here

IBM mainframes are likely to be the predominant legacy challenge and opportunity. However, there are other opportunities, like the IBM iSeries (previously known as the AS/400), and Unix servers. These additional, sometimes overlooked, server platforms represent a significant opportunity. Consider just the iSeries: there are hundreds of thousands of these boxes all over the world, with few vendors actually interfacing with them.

[1] Jim Sinur quoted on web site: http://java.sun.com/features/1999/08/unshackled1.html written by Janice J. Heiss

What Techniques are Available for Legacy Integration?

Rational Rapid Developer currently supports a variety of integration techniques. Different customers will certainly have different integration requirements, so the recommended integration technique will vary in most cases. Factors such as type of database, method of access, performance requirements, allowable code changes, and invasiveness all need to be assessed to determine the best solution recommended.

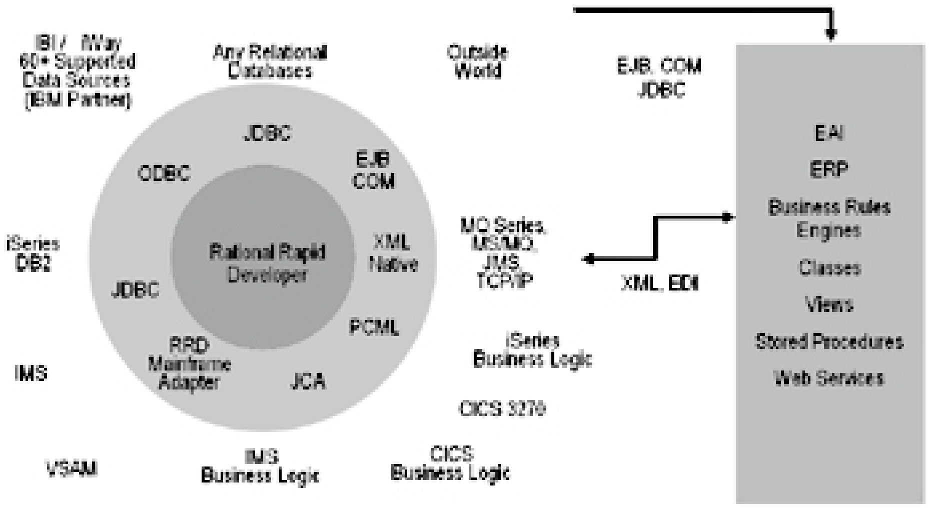


Figure 1. Legacy Integration Choices
A variety of integration techniques exist today, some within Rational Rapid Developer directly and others in partnership with third-party vendors. Table 1 displays the currently supported integration techniques:

Table 1. Legacy Integration Techniques

Integration	IBM Rational Rapid Developer	IBM Rational Rapid Developer Custom Method	IBM Rational Rapid Developer Mainframe Adapter	IBM Rational Rapid Developer JCA Adapter	IBM Rational Rapid Developer Messaging Adapter	iWay (Partner Adapter)
Relational databases	Yes					
VSAM files			Yes			Yes
IMS databases	Yes					Yes
IMS transactions				Yes		
IMS 3270 applications				Yes		
CICS programs				Yes		
CICS 3270 applications				Yes		
MVS or VM 3270 applications				Yes		
iSeries applications				Yes		
iSeries 5250 applications				Yes		
Messaging					Yes	
Message driven beans	Yes					
COM objects		Yes				
EJBs		Yes				
EAI		Yes			Yes	
ERP		Yes			Yes	
Rules engines		Yes			Yes	
Stored procedures	Yes	Yes				

Integration	IBM Rational Rapid Developer	IBM Rational Rapid Developer Custom Method	IBM Rational Rapid Developer Mainframe Adapter	IBM Rational Rapid Developer JCA Adapter	IBM Rational Rapid Developer Messaging Adapter	iWay (Partner Adapter)
Tables and views	Yes					
Web services	Yes					
Non-relational databases						Yes

How Do I Select the Proper Integration Technique?

Below are two charts to help in your integration technique selection. You need the following information:

- Type of legacy asset being accessed (Access)
- Where it is being accessed (Location)
- Method of access (Method)
- Performance requirement (Perform)
- Can developers insert additional logic into the system (Chg)
- Can they support the specified additional requirements (Requirements)

Step 1

In Table 2, locate the type of legacy asset being accessed and ensure that the details for that table row are acceptable for the situation.

Table 2. Legacy Asset Selection

Access	Vendor	Location	Method	Perform	Chg	Requirements
Relational Databases						
DB2 v4.5	IBM	iSeries	JDBC via DB2Connect	High	No	None
DB2 v5.1	IBM	iSeries	JDBC via DB2Connect	High	No	None
DB2 v6	IBM	Mainframe	JDBC via DB2Connect	High	No	DB2 server
DB2 v6	IBM	Middle tier server	JDBC direct access	High	No	None
DB2 v7	IBM	Mainframe	JDBC via DB2Connect	High	No	DB2 server
DB2 v7	IBM	Middle tier server	JDBC direct access	High	No	None
DB2 v8	IBM	Mainframe	JDBC via DB2Connect	High	No	DB2 server
DB2 v8	IBM	Middle tier server	JDBC direct access	High	No	None
Access	Microsoft®	Middle tier server	JDBC direct access	Low	No	None
SQL Server 7	Microsoft®	Middle tier server	JDBC direct access	High	No	None
SQL Server 2000	Microsoft®	Middle tier server	JDBC direct access	High	No	None
Oracle 7.3	Oracle	Middle tier server	JDBC direct access	High	No	None
Oracle 8i	Oracle	Middle tier server	JDBC direct access	High	No	None

Access	Vendor	Location	Method	Perform	Chg	Requirements
Oracle 9i	Oracle	Middle tier server	JDBC direct access	High	No	None
Sybase 11	Sybase	Middle tier server	JDBC direct access	High	No	None
Non-Relational Databases						
VSAM	Rational Rapid Developer Mainframe Adapter	CICS Mainframe	JDBC via DB2Connect	Medium	Yes	DB2 server
IMS DB	IBM	Mainframe	JDBC via IMS Java	High	No	IMS Java WebSphere zOS
Mainframe–Based Applications						
IMS transactions	Rational Rapid Developer JCA Adapter	IMS Mainframe	JCA via IMS Connect	Medium	Yes	IMS Connect
IMS 3270 applications	Rational Rapid Developer JCA Adapter	IMS Mainframe	JCA via Host on Demand	Medium	Yes	None
CICS programs	Rational Rapid Developer JCA Adapter	CICS Mainframe	JCA via CICS Transaction Gateway	Medium	Yes	CICS Transaction Gateway
CICS 3270 applications	Rational Rapid Developer JCA Adapter	CICS Mainframe	JCA via CICS Transaction Gateway	Medium	Yes	CICS Transaction Gateway
MVS or VM 3270 applications	Rational Rapid Developer JCA Adapter	Mainframe	JCA via Host on Demand	Medium	Yes	None
iSeries applications	Rational Rapid Developer JCA Adapter	iSeries	PCML via JTOpen	Medium	Yes	None
iSeries 5250 applications	Rational Rapid Developer JCA Adapter	iSeries	JCA via Host on Demand	Medium	Yes	None
Messaging (Rational Rapid Developer includes a Messaging Adapter feature)						
WebSphere MQ	IBM	iSeries	Natively accessed via API call	High	No	None
WebSphere MQ	IBM	Mainframe	Natively accessed via API call	High	No	None
WebSphere MQ	IBM	Middle tier server	Natively accessed via API call	High	No	None
MSMQ	Microsoft	Middle tier server	Natively accessed via API call	Medium	No	None
JMS	Any vendor that provides JMS	Middle tier server	Natively accessed via API call	Medium	No	None
Message-Driven Beans						
JMS	Any vendor that supports MDBs	Middle tier server	Natively accessed via API call	Medium	No	None
COM Objects (Callable if constructed to the Microsoft DNA platform)						

Access	Vendor	Location	Method	Perform	Chg	Requirements
COM application	Rational Rapid Developer Custom Method	Middle tier server	Custom Java method to invoke COM object	Medium	No	User must write custom method. Rational Rapid Developer provides sample code templates.
Enterprise Java Beans						
EJB application	Rational Rapid Developer Custom Method	Various	Custom Java method to invoke EJB	Medium	No	User must write custom method. Rational Rapid Developer provides sample code templates.
EAI, ERP and Rules Engines						
EAI	Any	Various	XML message, Java/J2EE API if provided by vendor	Medium	No	EAI system must accept XML messages from Rational Rapid Developer
ERP	Any	Various	XML message, Java/J2EE API if provided by vendor	Medium	No	ERP system must accept XML messages from Rational Rapid Developer
Rules engine	Any	Various	XML message, Java/J2EE API if provided by vendor	Medium	No	Rules engine must accept messages from Rational Rapid Developer
Stored Procedures						
Supported relational database	IBM Rational Rapid Developer Custom Method	Various	Custom Java method to invoke JDBC stored-procedure call	Medium	No	User must write custom method. Rational Rapid Developer provides sample code templates.
Tables and Views						
Supported relational database	Any	Various	Natively imported into class model through JDBC	High	No	None
Web Services						
SOAP	Industry standard	Various	Natively accessed via API call	Medium	No	None
UDDI	Industry standard	Various	Natively accessed via API call	Medium	No	None
Non-Relational Databases via iWay Software (IBM Partner) (Representative sample list here. You may use any non-relational iWay Software supported database.)						
Adabas	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
Btrieve	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
C-ISAM	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
Cloudbase	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter

Access	Vendor	Location	Method	Perform	Chg	Requirements
D-ISAM	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
DBMS	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
DBASE	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
FOCUS	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
IDMS	iWay Software	Mainframe	iWay ODBC/JDBC	High	No	iWay adapter
IMS	iWay Software	Mainframe	iWay ODBC/JDBC	High	No	iWay adapter
Informix	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
ISAM	iWay Software	Mainframe	iWay ODBC/JDBC	High	No	iWay adapter
MUMPS	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
PACE	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
Pick	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
QSAM	iWay Software	Mainframe	iWay ODBC/JDBC	High	No	iWay adapter
System 2K	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
Teradata	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
TOTAL	iWay Software	Various	iWay ODBC/JDBC	High	No	iWay adapter
VSAM	iWay Software	Mainframe	iWay ODBC/JDBC	High	No	iWay adapter

Step 2

In Table 3, locate the type of legacy asset being accessed and select the method that makes the most sense for the situation. In the left-hand column, scan for the legacy asset in which you are interested. Then scan to the right; for each environment that supports this type of legacy asset the performance throughput (Low, Medium or High) is shown. If your throughput requirements are identified, you can use that type of integration in your Rational Rapid Developer application.

Table 3. Legacy Method Selection

	IBM Rational Rapid Developer	IBM Rational Rapid Developer Custom Method	IBM Rational Rapid Developer Mainframe Adapter	IBM Rational Rapid Developer JCA Adapter	IBM Rational Rapid Developer Messaging Adapter	iWay (Partner Adapter)
Relational databases	High					
VSAM files			Medium			High
IMS databases	High					High
IMS transactions				Medium		
IMS 3270 applications				Medium		
CICS programs				Medium		
CICS 3270 applications				Medium		

	IBM Rational Rapid Developer	IBM Rational Rapid Developer Custom Method	IBM Rational Rapid Developer Mainframe Adapter	IBM Rational Rapid Developer JCA Adapter	IBM Rational Rapid Developer Messaging Adapter	iWay (Partner Adapter)
MVS or VM 3270 applications				Medium		
iSeries applications				Medium		
iSeries 5250 applications				Medium		
Messaging					High	
Message driven beans	Medium					
COM objects		Medium				
EJBs		Medium				
EAI		Medium			Medium	
ERP		Medium			Medium	
Rules engines		Medium			Medium	
Stored procedures	Medium	Medium				
Tables and views	High					
Web services	Medium					
Non-relational databases						High

How Do I Interface to Each Technique?

The scope of this document is to provide you a general assessment framework. For a more detailed technical description, refer to these Rational Rapid Developer documentation chapters in the online help:

- Database Modeling and Construction
- Message Modeling
- Mainframe Integration
- iSeries Integration

More Detail for Each Integration Technique

This section provides some level of detail for each integration technique.

Existing Relational Databases via Rational Rapid Developer

How does it work?

Rational Rapid Developer provides direct access to various relational databases, through a combination of design time and execution time techniques. At design time, you establish an ODBC driver interface to the data source to be used. By issuing ODBC interface calls for you, Rational Rapid Developer allows you to query the actual details of the relational database that this ODBC driver describes. Rational Rapid Developer automatically imports this database definition information into the application being created, saving a significant amount of setup time since all of the class, attribute and relationship information is captured automatically rather than manually entered.

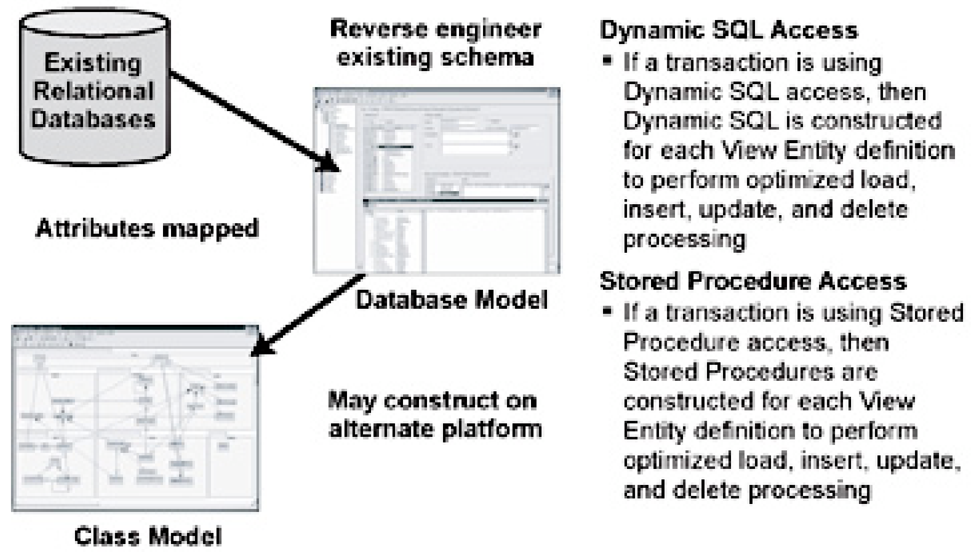

Existing Relational Databases
Reverse engineer existing schema
Attributes mapped
Database Model
Class Model
May construct on alternate platform
Dynamic SQL Access
If a transaction is using Dynamic SQL access, then Dynamic SQL is constructed for each View Entity definition to perform optimized load, insert, update, and delete processing
Stored Procedure Access
If a transaction is using Stored Procedure access, then Stored Procedures are constructed for each View Entity definition to perform optimized load, insert, update, and delete processing

Figure 2. Existing Relational Database at Execution Time

During application execution, Rational Rapid Developer uses the standard JDBC driver for its access to the data. The attributes of the JDBC driver will have been defined in the Rational Rapid Developer application, and it is through this JDBC driver that the actual rows and columns of the relational database are accessed.

Relational database access comes in two forms: via dynamic SQL statements, or through procedures stored within the database itself. Each of these techniques has its benefits, and the Rational Rapid Developer designer can determine which technique to use. If appropriate, each Web page in the application can use the most appropriate technique, which may differ from the technique used on other pages.

Why would you use it?

This is the "cleanest" form of relational database access, with seamless usage already built into the Rational Rapid Developer product. It's easy to set up and deploy, and was the basis for the design of the product. This should be your basis for all relational database access.

Additional Information

For more information about relational database integration, refer to current Rational Rapid Developer documentation.

Existing VSAM Files via the Rational Rapid Developer Mainframe Adapter

How does it work?

The Rational Rapid Developer Mainframe Adapter provides the ability to access mainframe VSAM (Virtual Storage Access Method) files running under CICS. It does this transparently to the Rational Rapid Developer application by making these environments "look" like SQL tables. Applications can issue SQL SELECT statements to load result sets containing specific VSAM data. They can also insert, delete and update VSAM records as easily as they can access a table row.

The IBM Rational Rapid Developer Mainframe Adapter is generated on the mainframe as a series of COBOL stored-procedure programs that are placed within the IBM DB2® environment. You require the mainframe DB2 server for this to work. These procedures react to the SQL statement and perform the appropriate functions.

The generation of these stored procedures is done based upon the user's own COBOL copy files that describe the various VSAM files. No changes to the user's code are required. A Rational Rapid Developer Mainframe Adapter generation program reads in any number of these COBOL copy files and generates the XML descriptors and COBOL stored procedures. These procedures are then compiled into the mainframe stored-procedure library, where they will be invoked at runtime when a Rational Rapid Developer application issues SQL calls.

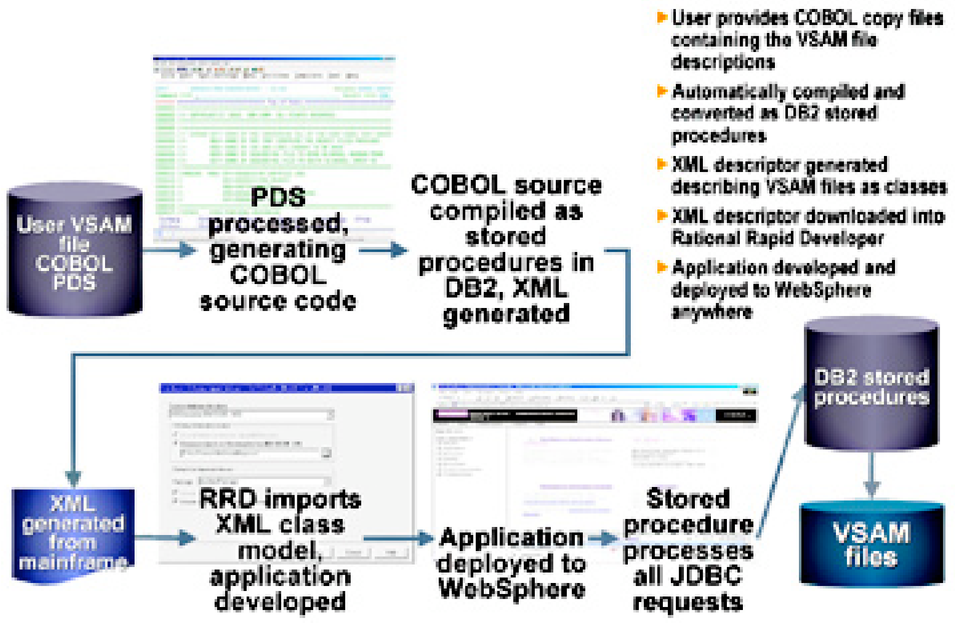

User provides COBOL copy files containing the VSAM file descriptions
Automatically compiled and converted as DB2 stored procedures
XML descriptor generated describing VSAM files as classes
XML descriptor downloaded into Rational Rapid Developer
Application developed and deployed to WebSphere anywhere
User VSAM file COBOL PDS
PDS processed, generating COBOL source code
COBOL source compiled as stored procedures in DB2, XML generated
DB2 stored procedures
XML generated from mainframe
RRD imports XML class model, application developed
Application deployed to WebSphere
Stored procedure processes all JDBC requests
VSAM files

Figure 3. Mainframe Adapter at Design and Execution Time

The user provides a source library containing all of the COBOL copy files that are needed to describe the mainframe VSAM files. Using this library as input, the Rational Rapid Developer Mainframe Adapter generation creates customizable source files that are compiled and linked into the DB2 stored-procedure library. These stored procedures interrogate incoming requests and set up an interface to the appropriate CICS subsystem where the actual processing occurs. As a byproduct of the generation, an XML source file is created describing the user environment. You download this file and import it into Rational Rapid Developer, simplifying setup. Once completed, a DB2 request can be made from the workstation through a DB2 Connect to the mainframe DB2 system. This invokes the generated program as a stored procedure within the stored-procedure address space. This address space is controlled by the mainframe Workload Manager environment, which allows the management of the environment including two-phase commit processing, when available.

Why would you use it?

This technique provides access to VSAM files, running under CICS on an IBM mainframe. It provides seamless access to mainframe systems through the IBM DB2 Connect facility. Although this approach is modest in cost, it is not intended for high-performance requirements.

Additional Information

See the Rational Rapid Developer documentation chapter titled: Mainframe Integration.

Existing IMS Databases via Rational Rapid Developer

How does it work?

Rational Rapid Developer provides direct access to IMS databases, through a combination of design time and execution time techniques. At design time, you upload and run a provided set of programs and JCL to parse existing COBOL copy files that describe your IMS environment. No changes to the user's code are required. A provided Rational Rapid Developer Mainframe Adapter generation program reads in any number of these COBOL copy files and automatically generates the source input to the IBM IMS Java DLIModel utility, which in turn generates a java class file that describes your IMS environment (for execution time) and an XMI file that gets downloaded to the workstation. Rational Rapid Developer imports this XMI file into the application being created, saving a significant amount of setup time since all of the class, attribute and relationship information is captured automatically rather than manually entered. At execution time, IBM IMS Java will make the IMS databases “look” like SQL tables and will respond to the Rational Rapid Developer generated SQL statements. Rational Rapid Developer application pages that use IBM IMS Java will be deployed to the mainframe WebSphere server.

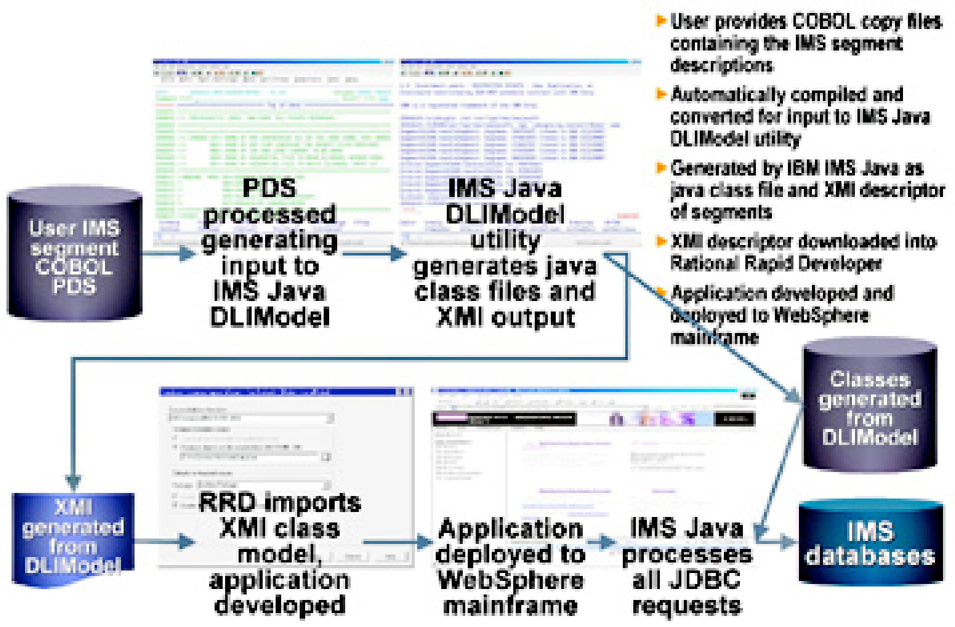

User provides COBOL copy files containing the IMS segment descriptions
Automatically compiled and converted for input to IMS Java DLIModel utility
Generated by IBM IMS Java as java class file and XMI descriptor of segments
XMI descriptor downloaded into Rational Rapid Developer
Application developed and deployed to WebSphere mainframe
User IMS segment COBOL PDS
PDS processed generating input to IMS Java DLIModel
IMS Java DLIModel utility generates java class files and XMI output
Classes generated from DLIModel
XMI generated from DLIModel
RRD imports XMI class model, application developed
Application deployed to WebSphere mainframe
IMS Java processes all JDBC requests
IMS databases

Figure 4. IMS Database at Design and Execution Time

The user provides a source library containing all of the COBOL copy files that are needed to describe the IMS database environment. Using this library as input, the Rational Rapid Developer Mainframe Adapter generation creates source input to the IBM IMS Java DLIModel utility. The DLIModel utility creates a java class file that describes the IMS environment for use at run time and an XMI source file describing the user environment. You download this XMI file and import it into Rational Rapid Developer, simplifying setup. Once completed, a SQL request can be made from WebSphere mainframe through IBM IMS Java to the mainframe IMS system.

Why would you use it?

This technique provides access to IMS hierarchical databases as if they were relational.

Additional Information

See the Rational Rapid Developer documentation chapter titled: Mainframe Integration.

IMS Transactions and CICS Programs via the Rational Rapid Developer JCA Adapter

How does it work?

Rational Rapid Developer provides direct access to IMS transactions and CICS programs, through a combination of design time and execution time techniques. At design time, you upload and run a provided set of programs and JCL to parse existing COBOL copy files that describe your IMS transaction I/O areas or your CICS DFHCOMM areas. No changes to the user’s code are required. A provided Rational Rapid Developer Mainframe Adapter generation program reads in any number of these COBOL copy files and automatically generates an XML file that gets downloaded to the workstation. When creating a “Legacy Service” within Rational Rapid Developer, you import this XML file into the application being created, saving a significant amount of setup time since all of the attribute information is captured automatically rather than manually entered. A custom method is then created within the Rational Rapid Developer application, which references the “Legacy Service”. At execution time, a JCA request is made to IMS through IMS Connect, or CICS through the CICS Transaction gateway.

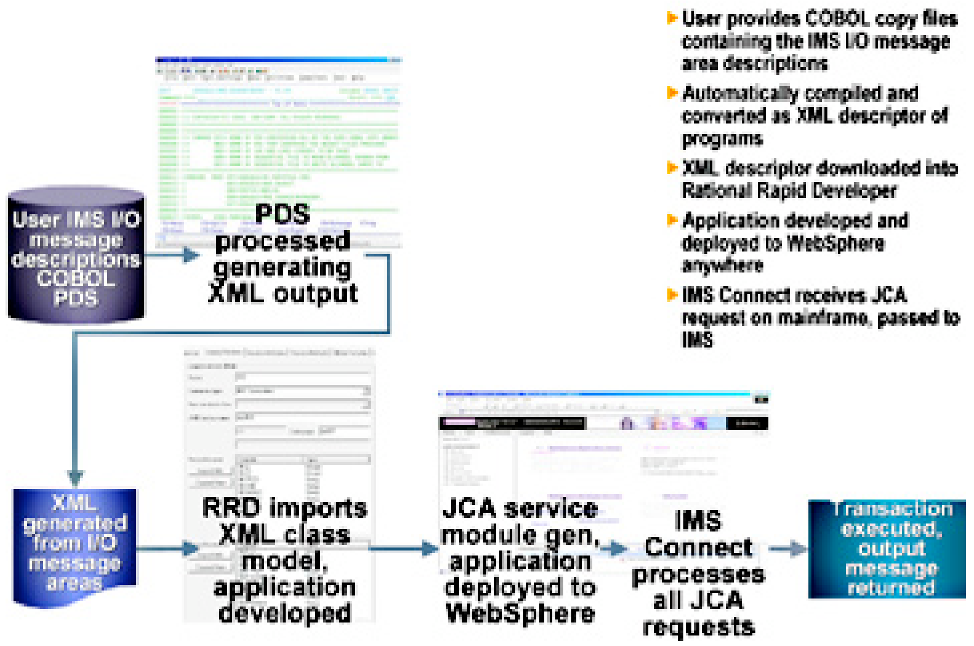

User IMS I/O message descriptions COBOL PDS
PDS processed generating XML output
XML generated from I/O message areas
RRD imports XML class model, application developed
JCA service module gen, application deployed to WebSphere
IMS Connect processes all JCA requests
Transaction executed, output message returned
User provides COBOL copy files containing the IMS I/O message area descriptions
Automatically compiled and converted as XML descriptor of programs
XML descriptor downloaded into Rational Rapid Developer
Application developed and deployed to WebSphere anywhere
IMS Connect receives JCA request on mainframe, passed to IMS

Figure 5a. IMS Transactions at Design and Execution Time

The user provides a source library containing all of the COBOL copy files that are needed to describe the IMS transaction. Using this library as input, the Rational Rapid Developer Mainframe Adapter generation creates an XML source file describing the user environment. You download this XML file and import it into Rational Rapid Developer, simplifying setup. Once completed, a JCA request can be made through IMS Connect to the mainframe IMS system.

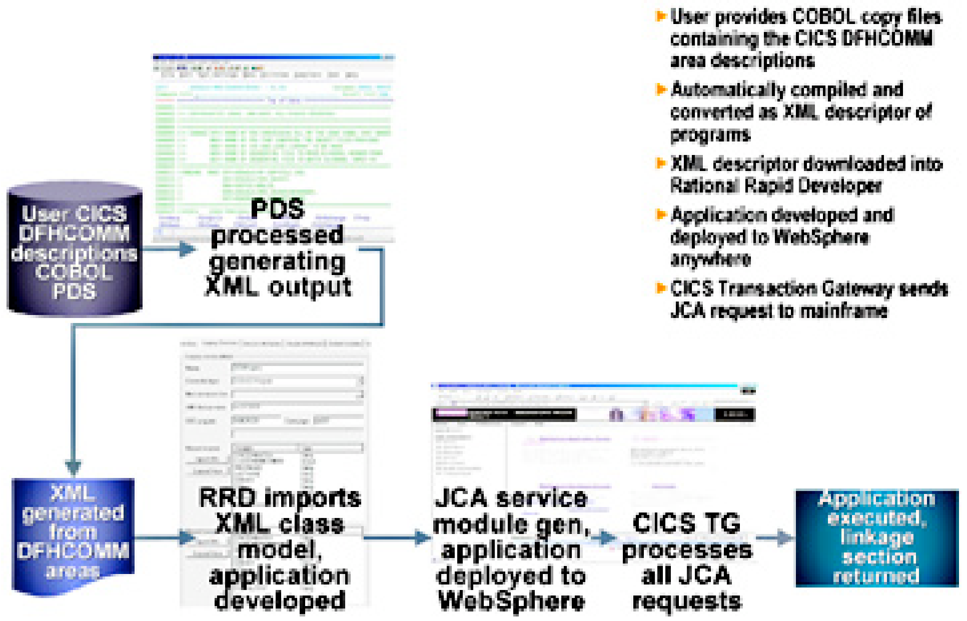

User CICS DFHCOMM descriptions COBOL PDS
PDS processed generating XML output
XML generated from DFHCOMM areas
RRD imports XML class model, application developed
JCA service module gen, application deployed to WebSphere
CICS TG processes all JCA requests
Application executed, linkage section returned
User provides COBOL copy files containing the CICS DFHCOMM area descriptions
Automatically compiled and converted as XML descriptor of programs
XML descriptor downloaded into Rational Rapid Developer
Application developed and deployed to WebSphere anywhere
CICS Transaction Gateway sends JCA request to mainframe

Figure 5b. CICS Programs at Design and Execution Time

The user provides a source library containing all of the COBOL copy files that are needed to describe the CICS program. Using this library as input, the Rational Rapid Developer Mainframe Adapter generation creates an XML source file describing the user environment. You download this XML file and import it into Rational

Rapid Developer, simplifying setup. Once completed, a JCA request can be made through the CICS Transaction Gateway to the mainframe CICS system.

Why would you use it?

This technique provides access to IMS transactions or CICS programs, through the IBM standard JCA connector.

Additional Information

See the Rational Rapid Developer documentation chapter titled: Mainframe Integration.

CICS 3270 Applications via the Rational Rapid Developer JCA Adapter

How does it work?

Rational Rapid Developer provides direct access to CICS 3270 applications, through a combination of design time and execution time techniques. At design time, you upload and run a provided set of programs and JCL to parse existing CICS load libraries. In a read-only fashion, these CICS load libraries are scanned for BMS mapsets. For each mapset located, parsing is done against it and XML is created that gets downloaded to the workstation. No changes to the user's code are required. When creating a "Legacy Service" within Rational Rapid Developer, you import this XML file into the application being created, saving a significant amount of setup time since all of the attribute information is captured automatically rather than manually entered. A custom method is then created within the Rational Rapid Developer application, which references the "Legacy Service". At execution time, a JCA request is made to CICS through the CICS Transaction gateway.

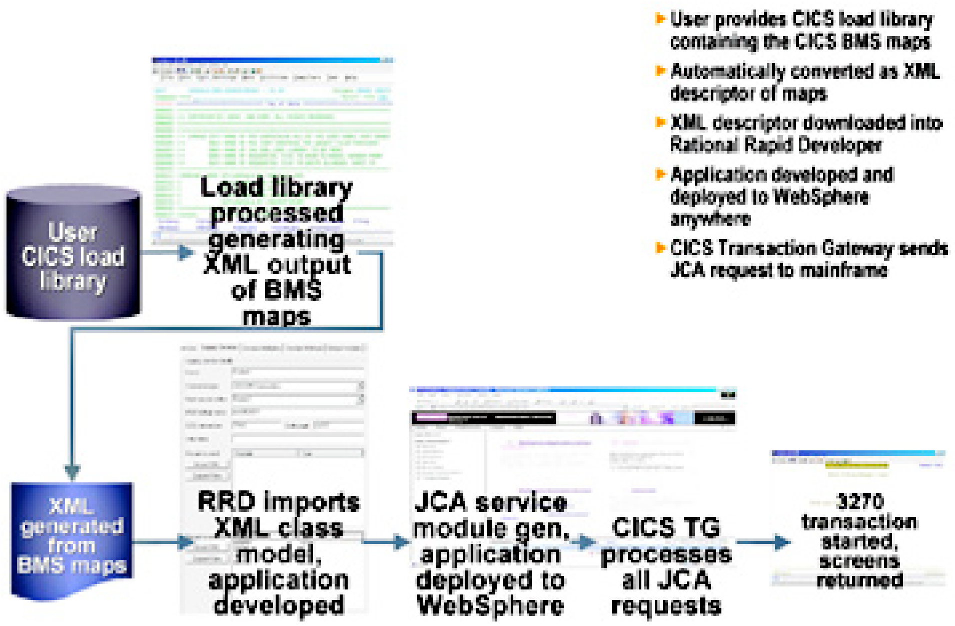

User provides CICS load library containing the CICS BMS maps
Automatically converted as XML descriptor of maps
XML descriptor downloaded into Rational Rapid Developer
Application developed and deployed to WebSphere anywhere
CICS Transaction Gateway sends JCA request to mainframe
User CICS load library
Load library processed generating XML output of BMS maps
XML generated from BMS maps
RRD imports XML class model, application developed
JCA service module gen, application deployed to WebSphere
CICS TG processes all JCA requests
3270 transaction started, screens returned

Figure 6. CICS 3270 Applications at Design and Execution Time

The user provides a CICS load library containing all of the BMS mapsets that are needed to describe the CICS 3270 application. Using this library as input, the Rational Rapid Developer Mainframe Adapter generation creates an XML source file describing the user environment. You download this XML file and import it into Rational Rapid Developer, simplifying setup. Once completed, a JCA request can be made through the CICS Transaction Gateway to the mainframe CICS system.

Why would you use it?

This technique provides access to one or a series of CICS 3270 applications, through the IBM standard JCA connector.

Additional Information

See the Rational Rapid Developer documentation chapter titled: Mainframe Integration.

IMS 3270, MVS or VM 3270, or iSeries 5250 Applications via the Rational Rapid Developer JCA Adapter

How does it work?

Rational Rapid Developer provides direct access to IMS 3270, MVS or VM 3270, or iSeries 5250 applications, through a combination of design time and execution time techniques. At design time, you start up your 3270/5250 session as well as the Rational Rapid Developer EHLLAPI Collector. As you flow through your screens for the desired application, you collect the 3270/5250 details with the EHLLAPI collector. When finished, the EHLLAPI collector will write out an XML file. No changes to the user's code are required. When creating a "Legacy Service" within Rational Rapid Developer, you import this XML file into the application being created, saving a significant amount of setup time since all of the attribute information is captured automatically rather than manually entered. A custom method is then created within the Rational Rapid Developer application, which references the "Legacy Service". At execution time, a JCA request is made to the desired system through IBM's Host on Demand.

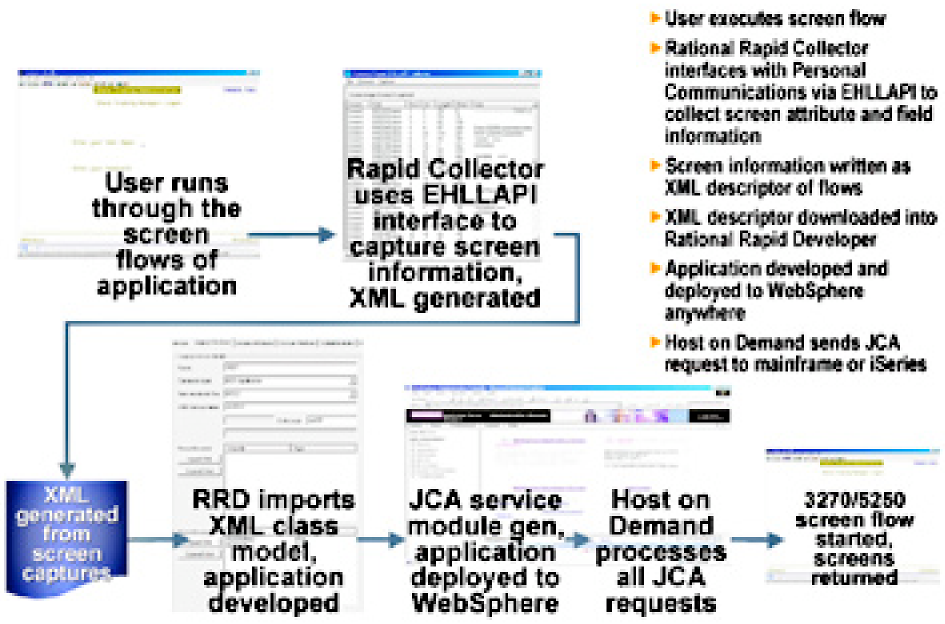

User executes screen flow
Rational Rapid Collector interfaces with Personal Communications via EHLLAPI to collect screen attribute and field information
Screen information written as XML descriptor of flows
XML descriptor downloaded into Rational Rapid Developer
Application developed and deployed to WebSphere anywhere
Host on Demand sends JCA request to mainframe or iSeries
User runs through the screen flows of application
Rapid Collector uses EHLLAPI interface to capture screen information, XML generated
XML generated from screen captures
RRD imports XML class model, application developed
JCA service module gen, application deployed to WebSphere
Host on Demand processes all JCA requests
3270/5250 screen flow started, screens returned

Figure 7. IMS 3270, MVS or VM 3270, or iSeries 5250 Applications at Design and Execution Time

The user flows through the desired transaction, capturing the screen details with the Rational Rapid Developer EHLLAPI Collector. The Rational Rapid Developer EHLLAPI Collector creates an XML source file describing the user environment. You import this XML file into Rational Rapid Developer, simplifying setup. Once completed, a JCA request can be made through the IBM Host on Demand connector to the desired system.

Why would you use it?

This technique provides access to one or a series of IMS 3270, MVS or VM 3270, or iSeries 5250 applications, through the IBM standard JCA connector.

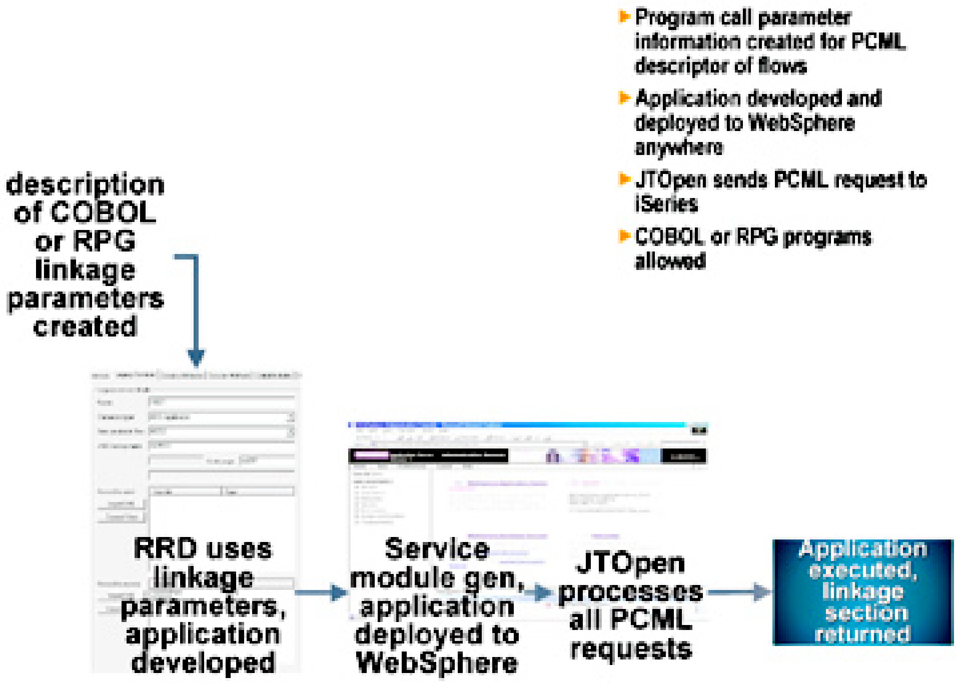
Additional Information

See the Rational Rapid Developer documentation chapter titled: Mainframe Integration.

iSeries Applications via the Rational Rapid Developer PCML Adapter

How does it work?

Rational Rapid Developer provides direct access to iSeries applications, either COBOL or RPG, through a combination of design time and execution time techniques. At design time, you define a legacy service that will talk to the iSeries. Associated with this legacy service are the input and output parameters related to the call. These parameters are defined using the edit feature of the legacy service. This allows you to define the input and output linkage section parameters to either COBOL or RPG iSeries applications. No changes to the user's code are required. A custom method is then created within the Rational Rapid Developer application, which references the "Legacy Service". At execution time, a PCML request is made to the desired system through the JTOpen call within the iSeries jt400.jar.



Program call parameter information created for PCML descriptor of flows
Application developed and deployed to WebSphere anywhere
JTOpen sends PCML request to iSeries
COBOL or RPG programs allowed
description of COBOL or RPG linkage parameters created
RRD uses linkage parameters, application developed
Service module gen, application deployed to WebSphere
JTOpen processes all PCML requests
Application executed, linkage section returned

Figure 8. iSeries Applications at Design and Execution Time

The user provides the input and output linkage section parameters through the edit feature associated with the legacy service. The edit feature allows you to define the input and output fields associated with the call. Once completed, a PCML request can be made through the JTOpen call within the iSeries jt400.jar to the desired system.

Why would you use it?

This technique provides access to any COBOL or RPG iSeries application program through the IBM standard JTOpen connector.

Additional Information

See the Rational Rapid Developer documentation chapter titled: iSeries Integration.

Messaging Systems via the Rational Rapid Developer Messaging Adapter

How does it work?

The Rational Rapid Developer Messaging Adapter allows the customer to model and deploy XML and non-XML messages through a variety of transport techniques, communicating with various message destinations, both JMS and non-JMS. This is key if your application needs to integrate with heterogeneous transports. Messages can be inbound or outbound, have associated custom methods, and can be used to communicate with other systems like EAI, ERP and rules engines, to name a few.

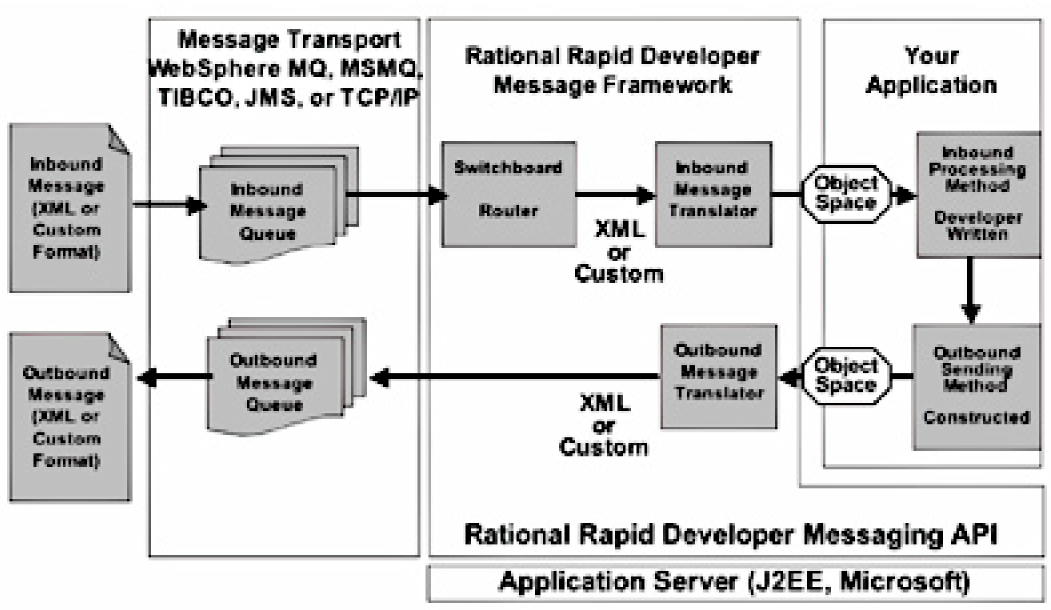

Message Transport
WebSphere MQ, MSMQ, TIBCO, JMS, or TCP/IP
Rational Rapid Developer Message Framework
Your Application
Inbound Message (XML or Custom Format)
Inbound Message Queue
Switchboard Router
Inbound Message Translator
Object Space
Inbound Processing Method Developer Written
XML or Custom
Outbound Message (XML or Custom Format)
Outbound Message Queue
Outbound Message Translator
Object Space
Outbound Sending Method Constructed
XML or Custom
Rational Rapid Developer Messaging API
Application Server (J2EE, Microsoft)

Figure 9. Messaging Adapter at Execution Time

Messages are defined within Rational Rapid Developer as either input or output, and then modeled. This modeling allows you to import a DTD, in the case of XML, and associate the DTD elements to Rational Rapid Developer attributes via a simple drag and drop. This association then allows these messages to utilize the Rational Rapid Developer messaging framework to its full function. You can optionally develop custom methods, with the help of the Rational Rapid Developer code editor and code pattern system, and connect these methods to an inbound or outbound message. The Rational Rapid Developer Messaging Adapter runs asynchronously on the middle-tier server, and can run without intervention from the end-user.

Why would you use it?

The Rational Rapid Developer Messaging Adapter is the supported gateway to all messaging systems, both XML-based and otherwise. ERP and EAI systems can be accessed through this technique. This is the “cleanest” messaging system access, with seamless usage already built into Rational Rapid Developer. It’s easy to set up and deploy, and was the basis for the design of the product. This should be your basis for all messaging system access.

Additional Information

See the Rational Rapid Developer documentation chapter titled: D.2 Message Modeling.

Custom Methods via Rational Rapid Developer

How does it work?

One of the most valuable features of Rational Rapid Developer is that it allows you to ignore the coding requirements for any of the highly complex infrastructure code required for *n*-tier deployments, and instead be concerned only with the actual business logic code snippets that make the Web site unique to the customer. This saves a significant amount of programming time, and allows developers whose skill set and experience do not include Web development to be highly productive in these new technologies.

Because of this, Rational Rapid Developer allows developers to write and invoke specific business logic as custom Java™ methods. Numerous inflection points are provided that allow developers to insert code at exactly the right location: methods connected to an action like a page button, methods to validate or initialize an attribute, or methods to override existing database calls, for example. Developers will code some useful method logic that allows them to unleash the entire power of the Java runtime libraries, if desired.

Some integrations will require custom methods to be written. We understand that certain specific logic will need to be written, and through the use of the code editor and available code pattern system, Rational Rapid Developer helps developers write these methods.

Why would you use it?

Custom methods allow application developers to write code to any unsupported systems that are deemed necessary. These all take the form of Java code deployed in the middle tier. Rational Rapid Developer provides most of the infrastructure code, which allows the application developer to concentrate, unhindered, on the actual business logic code. This should be your basis for all unsupported system access.

Message-Driven Beans via Rational Rapid Developer

How does it work?

Rational Rapid Developer can construct message-driven beans (MDBs) to process and handle incoming messages, via JMS (Java Messaging Service), interacting with the middle-tier server instead of the Rational Rapid Developer Messaging Adapter. They work in the same techniques as described above for the Messaging Adapter.

Rational Rapid Developer MDB support requires a J2EE application server that supports message-driven beans, and uses JMS. This facility allows the customer to model and deploy XML and non-XML messages through JMS, to and from various message destinations. Messages can be inbound or outbound, have associated custom methods, and can be used to communicate with other systems like EAI, ERP and rules engines.

Why would you use it?

If your environment includes a J2EE application server that supports MDBs, you would use this Rational Rapid Developer facility for the same reasons as the Rational Rapid Developer Messaging Adapter.

Web Services via Rational Rapid Developer

How does it work?

Built-in to Rational Rapid Developer is the Web service interface that will allow developers to both discover existing Web services and to publish new ones. These Web services are available to the developer from any of the custom methods they write. The code editor helps with the selection and usage of the previously discovered Web services. API calls to invoke the Web service are automatically handled by Rational Rapid Developer.

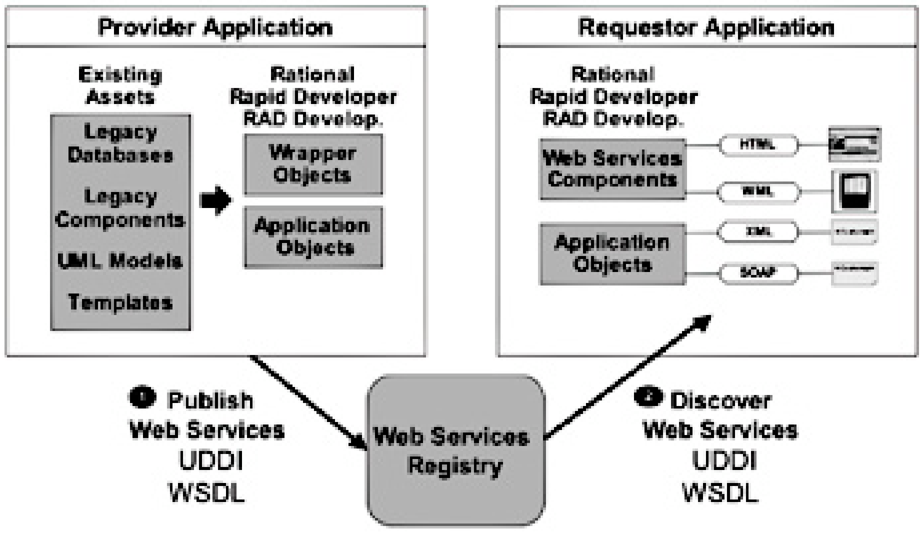


Figure 10. Web Services at Execution Time

Rational Rapid Developer helps developers publish any desired Web services. Once again, API calls to publish the Web service are handled automatically by Rational Rapid Developer.

Why would you use it?

Web services allow application developers to discover and access previously published application code, anywhere on the Internet, and to optionally publish their own useful routines for others to use. Rational Rapid Developer provides most of the infrastructure code, which allows the application developer to concentrate, unhindered, on the actual business logic code. This should be your basis for all Web application program access.

Existing Non-Relational Databases via iWay™ Software (IBM Partner)

How does it work?

iWay Software (a wholly-owned company of Information Builders Inc.) provides both sides of the communication interface to allow support for a large variety of data sources. The customer installs one or more iWay servers associated with one or more data sources. At design-time, you set up an iWay ODBC/JDBC driver interface to the intended target data source. By issuing ODBC interface calls for you, Rational Rapid Developer allows you to query the actual details of the non-relational database with which this ODBC driver is associated. Through the iWay, these details appear to be relational in nature. Rational Rapid Developer automatically imports this information into the application being created, saving a significant amount of setup time since all of the class, attribute and relationship information is captured automatically rather than manually entered.

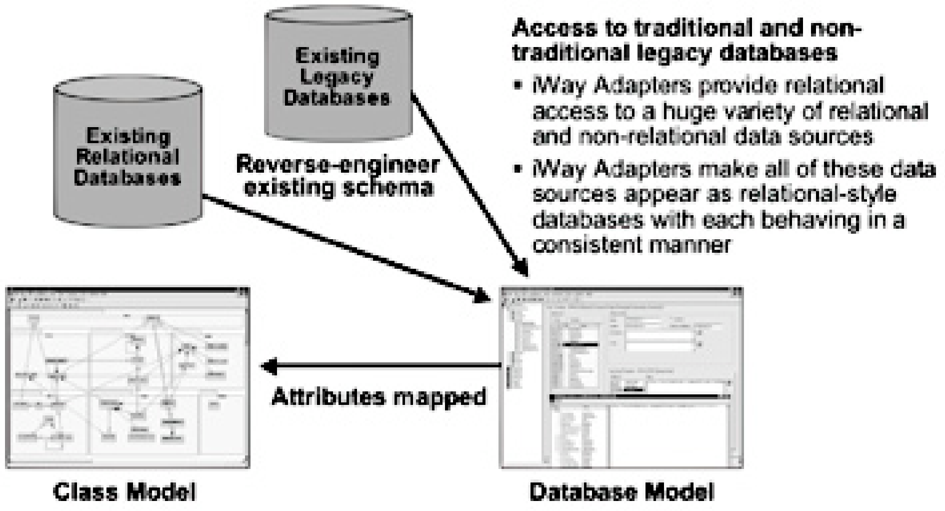

Existing Legacy Databases
Existing Relational Databases
Reverse-engineer existing schema
Access to traditional and non-traditional legacy databases
• iWay Adapters provide relational access to a huge variety of relational and non-relational data sources
• iWay Adapters make all of these data sources appear as relational-style databases with each behaving in a consistent manner
Attributes mapped
Class Model
Database Model

Figure 11. Databases via iWay Adapter at Execution Time

At execution time, Rational Rapid Developer uses the supplied iWay JDBC driver. The attributes of the JDBC driver will have been defined in the Rational Rapid Developer application, and it is through this JDBC driver that the actual simulated rows and columns of the non-relational database are accessed. Since iWay support for all iWay adapters is generally coded to a specific common baseline, this may cause various restrictions when using some of the available data sources. Users might not be able to take advantage of some features in a specific database, as all iWay adapters do not share in this support.

Why would you use it?

iWay Adapters provide access to all of those other, usually legacy, database engines, well-known or otherwise. Simply put, if iWay does not support it, it probably doesn't exist. iWay Adapters provide access to both relational and non-relational databases. However it is only recommended to use the non-relational support to legacy systems, as various features of the relational databases have been excluded in the iWay support.

Who Can I Contact for Additional Information?

Contact me, Jeff Douglas, (jeffdouglas@us.ibm.com), Advisory Software Engineer, or James Farrell, (jrfarrel@us.ibm.com) Rational Rapid Developer, Product Manager for any additional help. We will be happy to oblige.

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.