

Integrate Green-screen Applications in your SOA: Using Rational Host Access Transformation Services (HATS)

An overview of HATS Web service support

Skill Level: Introductory

[Rick Hardison \(rhardison@sdicorp.com\)](mailto:rhardison@sdicorp.com)
Consultant
SDI Corp.

09 Apr 2009

This article summarizes the support provided by IBM® Rational® Host Access Transformation Services (HATS) that allows you to create Web services to provide standard programming interfaces to business logic and transactions contained within character-based 3270 and 5250 applications (also referred to as host applications or green-screen applications).

The Challenge

Host applications continue to be a significant part of the mix of the business applications within a company. However, business applications cannot remain static. They have to be changed to meet business needs and to adapt to technology advancements.

Businesses are looking for ways to leverage their IT assets, while looking for alternatives that save money and reduce the strain on already scarce skills. This includes the challenge of looking for ways to transform host applications to integrate with new and existing business applications more easily.

One possible solution is to use Web services technology as a common denominator over which applications of different technologies can communicate and be integrated

together in a Service Oriented Architecture (SOA) environment.

Solutions

IBM Enterprise Modernization solutions help organizations cost-effectively and incrementally evolve core IT systems towards modern architectures and technologies - reducing maintenance burden and freeing up more of their resources to focus on developing new business requirements and capabilities.

Architecture modernization solutions include:

- Rational Developer for System z
- Rational Developer for i
- Rational Developer for i for SOA Construction
- Rational Business Developer
- Rational Software Architect
- Rational Host Access Transformation Services
- IBM Data Studio
- IBM Problem Determination Tools

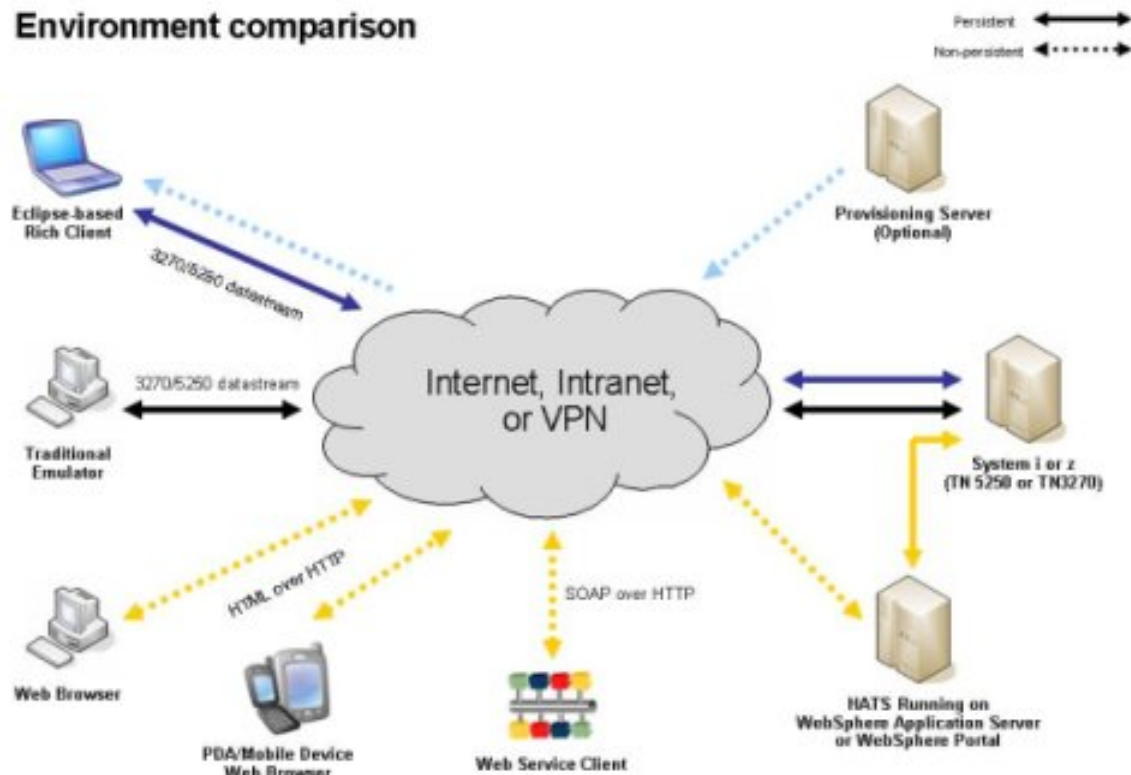
HATS is best known for transforming 3270 and 5250 host applications and delivering them as Web or rich client applications. But, HATS can also be used to create Web services to provide standard programming interfaces to business logic and transactions contained within 3270 and 5250 host applications as part of a SOA. HATS Web services can also provide access to data from virtual terminal (VT) emulation screens.

HATS Overview

Figure 1 below illustrates various methods for accessing host applications. With a traditional emulator, users access host applications over 3270 or 5250 sessions, and are presented with the traditional character-based (green-screen) host application interface. With HATS Web support, users access host applications using HTML over HTTP through Web browsers, or using SOAP over HTTP through Web services clients, and by way of a HATS application running on WebSphere Application Server. With HATS rich client support, users access host applications over 3270 or 5250 sessions through the HATS rich client application running on their workstations. With HATS, users are presented with an easy-to-use GUI instead of the traditional character-based interface.

Figure 1. Accessing host applications

Environment comparison



In addition to supporting TN3270, TN3270E, and TN5250 session protocols, HATS supports access to 5250 applications through a WebFacing server which does not require online transaction processing (OLTP) capacity (interactive CPW). This significantly reduces the overall cost of using HATS with 5250 applications by removing the interactive feature requirement associated with TN5250 session connectivity. HATS also supports programmatic access (described below) to virtual terminal (VT) host applications.

HATS has two components, the HATS Toolkit and the HATS runtime.

You create and test a HATS Web, portlet, mobile, or rich client application using the HATS Toolkit, then deploy the HATS application including the HATS runtime to a WebSphere Application Server, a WebSphere Portal, or a rich client platform for access by the user community or consuming application.

The HATS Toolkit plugs into the Eclipse-based IBM Rational Software Delivery Platform, hereafter referred to as the “SDP”. The HATS Toolkit provides a development environment for creating HATS applications. HATS applications provide presentation and programmatic interfaces to host applications non-invasively (no changes required to the host application).

Presentation transformation includes the ability to recognize host screens and transform them in real time to GUI pages according to a set of predefined rules. The rules can easily be modified to match the requirements of the host application. Screen components can be transformed to elements such as drop down lists, hot links, tables, buttons, valid value lists, tabbed folders, and graphs – giving users plenty of tools for navigating the host application.

Programmatic function for HATS includes macro support that allows for programmed navigation through multiple host screens. HATS support for global variables allows the use of powerful programming techniques. Switches can be set, or data collected, while traversing one area of the host application and tested, or used, in another. Programmatic support can be combined with presentation transformation to further improve the user's experience with the host application. Programmatic support is also the basis of HATS ability to collect data from multiple back end systems and to provide Web services interfaces for host applications.

The remainder of this article describes how HATS can be used easily and non-invasively to develop a Web service interface to a host application.

Key Terms

Some of the basic terms and concepts related to HATS Web services support are:

Macro – An XML script that defines a set of screens and actions that should be taken on those screens. Macros can be recorded and played to automate user interactions with the host. Macros can be used to skip screens, loop, prompt users for data input, insert data from global variables, and extract host screen information.

Integration Object – An Integration Object, or IO, is a Java class that encapsulates a macro. An Integration Object class has a method that runs the macro.

Integration Object chaining – Integration Object chaining enables creation of multiple Integration Objects, grouped together to perform a single major task. Each Integration Object performs one subtask, and the major task is performed by the Integration Object chain. When the chain is run, the Integration Objects run in sequence, each using the same host connection.

Prompt – A prompt is a screen-level action that inserts some value into an input field on a host screen. Prompt values can be provided from a user (or a caller of the Web service operation) or from a calculated value. A prompt in a macro results in an input parameter to a Web service operation.

Extract – An extract is a screen-level action that extracts the value of a host field or of a region. An extract in a macro results in an output parameter to a Web service operation.

HATS Web service Implementation Overview

The following is an overview of how to use the HATS Toolkit to create a Web service that accesses data from a host application. Recorded and live demos can be found at http://rational.demos.ibm.com/atdemo/atdemo_hats.html. Click on the Web services link.

Figure 2 summarizes the steps in creating a HATS Web service. First a HATS project is created. While creating a HATS project, the connection to the backend host site is defined. Then HATS macros are created that are used to connect to and navigate through the host application. Parameters such as host session connection pooling are set up for the connection to the host application. HATS Integration Objects (IOs) are created from one or more of the macros. The IOs provide a programming interface to execute the macros. From the IO, or IOs, a Web service is created and tested using the Web Services Explorer that is built into the SDP. Finally, a sample client that invokes the Web service is created and tested.

Figure 2. HATS Web services implementation overview

HATS Toolkit

- 1. Create HATS project
Setup development environment
Define destination host



- 2. Create macros
Navigate through host screens
Connect, Data, Disconnect macros



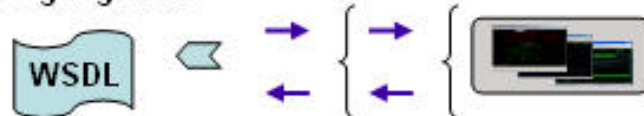
- 3. Set connection parameters
Host session connection pool
Connect and Disconnect macros



- 4. Create Integration Object
Programming interface for Data macros



- 5. Create and test Web service
Web Services support files
Web Services Definition Language File



- 6. Create and test Web service client
From WSDL File

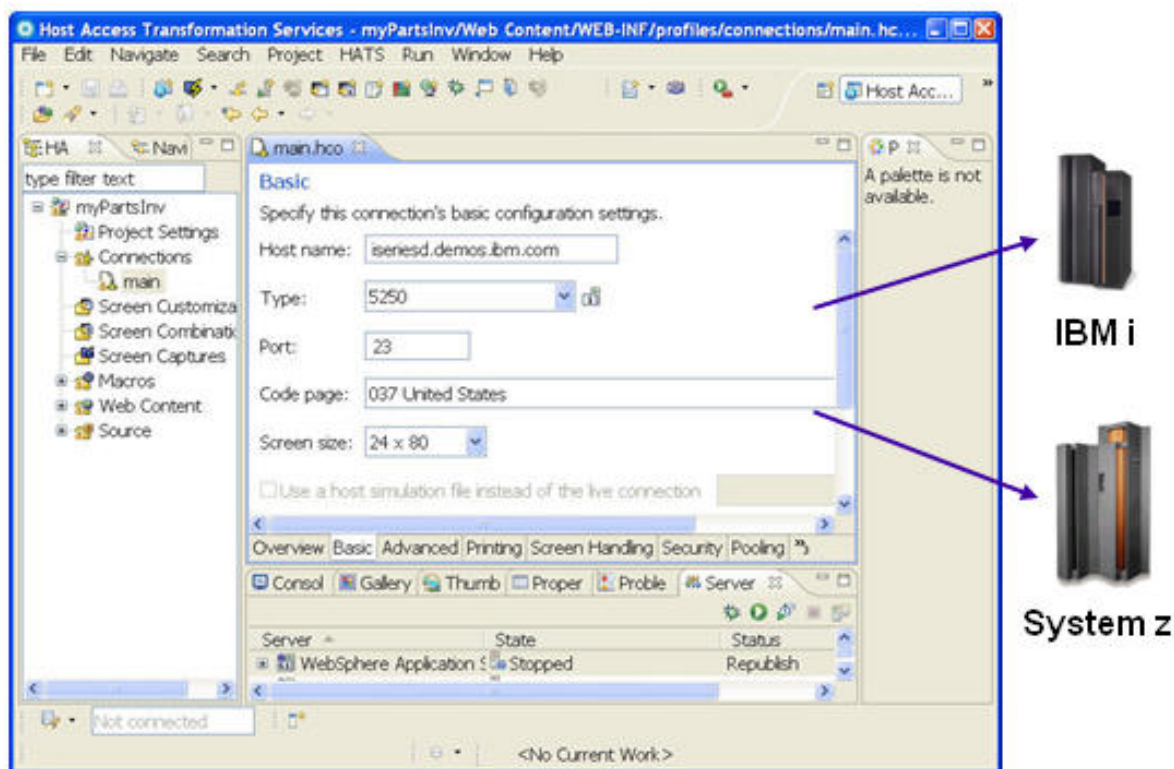


Create a HATS project

Within the SDP, HATS development efforts are organized by project. From a project, a HATS application is generated that can be deployed to run on WebSphere Application Server.

When creating a HATS project, a key component in the project's settings is the definition of the connection to the backend host system. HATS applications can communicate with one or more backend host systems. Required settings include the TCP/IP host name and port of the Telnet server running on the target backend host and the protocol to be used (for example 5250 or 3270).

Figure 3. HATS project connection settings



Create macros

As mentioned earlier, HATS applications provide non-invasive Web presentation and programmatic interfaces to host applications. When creating a HATS application that provides a Web services interface for a host application, HATS programmatic functions are used. A key element of HATS programmatic function is macro support. Macros provide programmed navigation through multiple host screens.

It is easy to create macros using the HATS Toolkit. You can record macros using a wizard as you navigate through host screens using a live host connection. By default, screen captures are automatically created for every screen navigated while

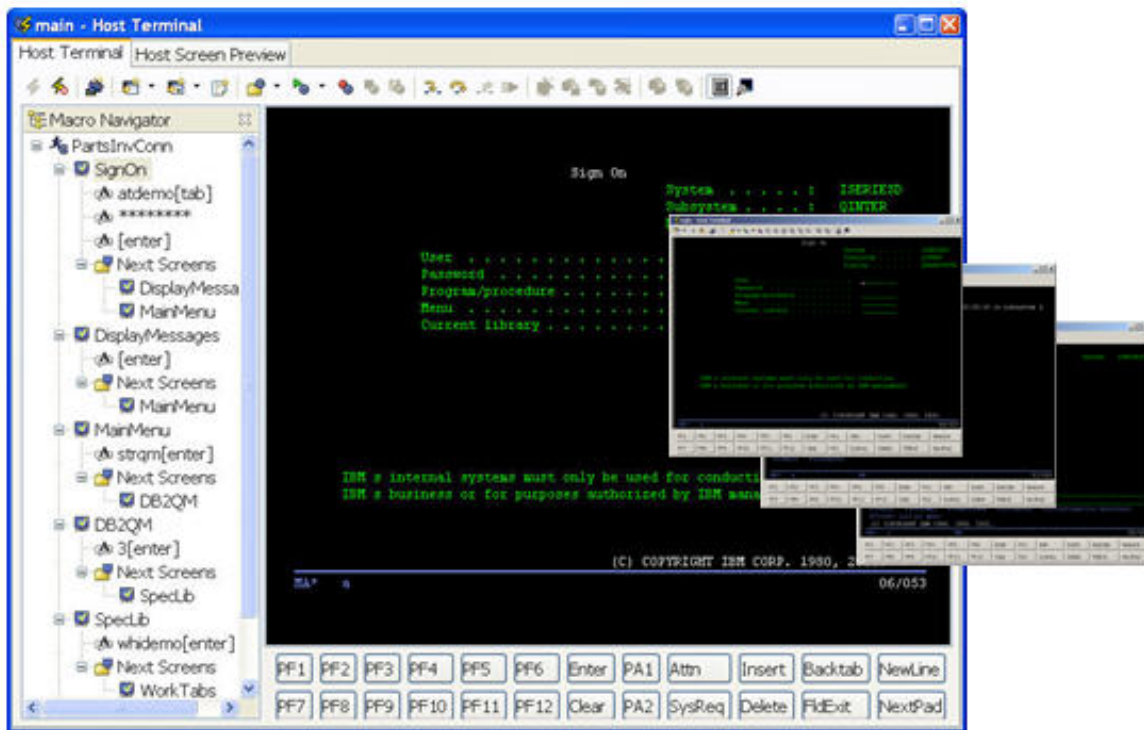
recording a macro. These screen captures can be used within the Visual Macro Editor if you must later make changes to the macro.

To create a Web service interface for a host application, at least one macro is required. We will call this the Data macro. If you want to improve performance, you can use connection pooling (discussed below). Two macros, Connect and Disconnect, are recommended for this.

Connect macro

Host 5250 and 3270 applications are session oriented. To communicate, the HATS application must create a session connection with the host application. The Connect macro creates the connection and primes it by navigating to the point where the Data macro can begin.

Figure 4. Connect macro



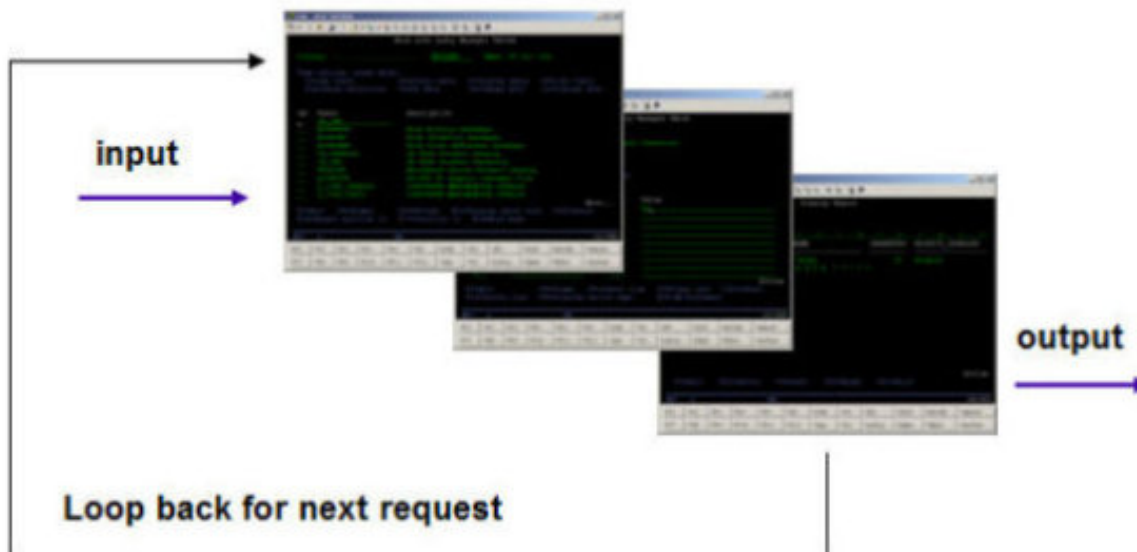
Data macro

Macro prompts and extracts can optionally be used to provide input to and supply output from macros.

The purpose of the Data macro in this example is to prompt for input, navigate through the application to extract data based on the requested input, and then navigate back to where it can prompt for the next input. If a Connect macro is used

in conjunction with connection pooling, a connection is reused over and over again by multiple executions of the Data macro. The Web service is derived from the Data macro.

Figure 5. Data macro



Disconnect macro

In conjunction with connection pooling, use the Disconnect macro to sign off the host system and end the connection between the HATS and host applications.

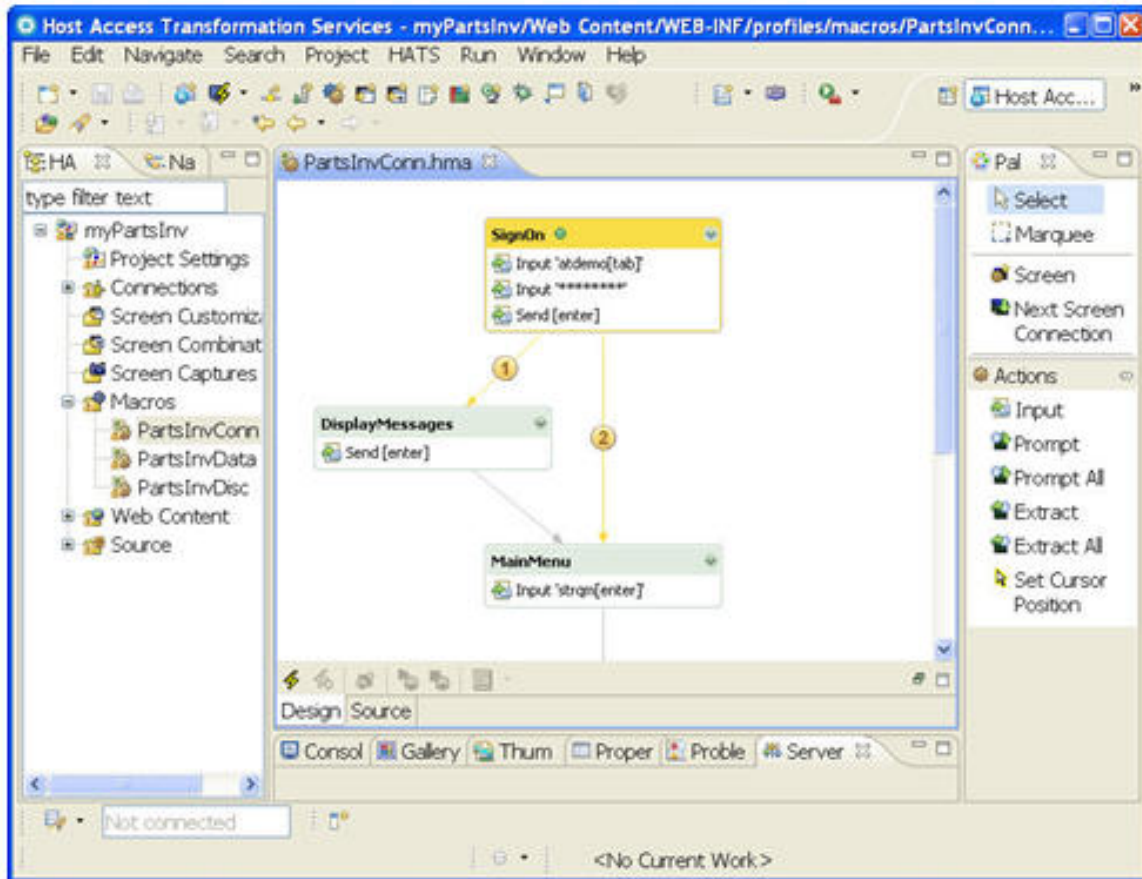
Visual Macro Editor

The Visual Macro Editor (VME) is a tool for visually developing HATS macros enabling you to build robust macros more easily and to find logic problems within macros more easily at development time.

The VME combines many of the features of the HATS host terminal, basic Macro Editor, and Advanced Macro Editor into one tool, and allows for offline development of macros. It allows flows to be copied between macros and provides drag-and-drop support for adding new screens.

The VME is particularly useful for adding conditional paths through a macro, for example, to include an error path, and for adding or changing screen-level recognition criteria and actions.

Figure 6. HATS Visual Macro Editor



Set connection parameters

A HATS application uses a connection (a 5250 or 3270 session) to communicate with the host application. Basic configuration parameters for the connection, for example the TCP/IP host name of the Telnet server for the host application, are set when the project is first built. More connection parameters can also be defined, including connection pooling and which Connect and Disconnect macros to use for the connection.

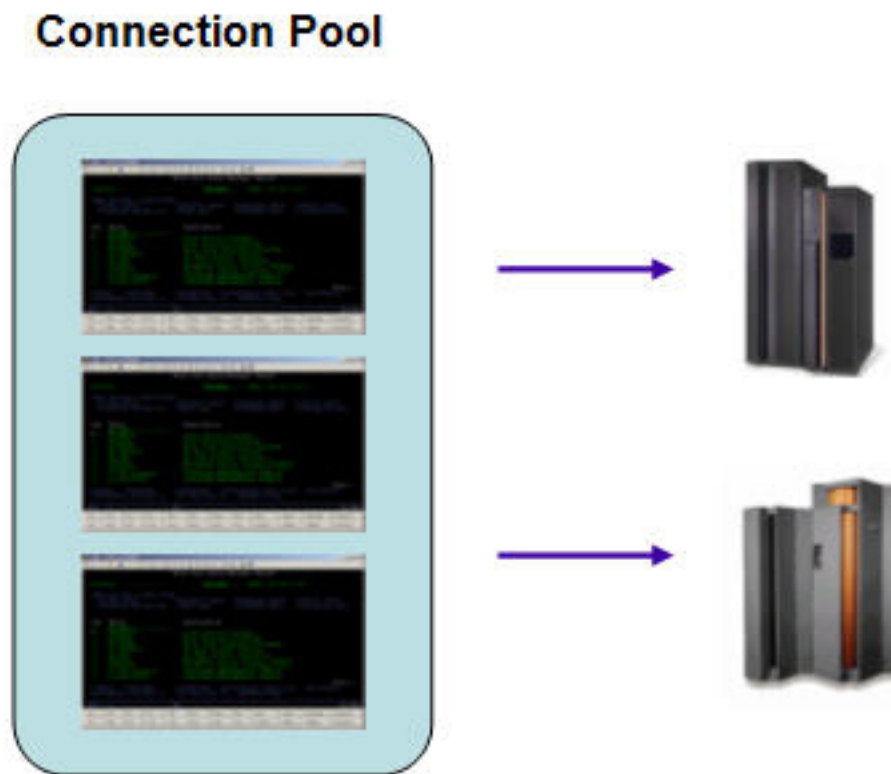
Connection pooling

HATS uses a function called connection pooling to improve response time for the Web services client and performance for both the HATS application providing the Web service and the host application. Connection pooling allows a number of connections (5250 or 3270 sessions) to be specified that HATS maintains in a pool already connected and ready to be used by the HATS application.

When used in conjunction with Connect and Disconnect macros, HATS maintains a number of primed connections in the pool by running the Connect macro. Each of

the connections is ready to run the Data macro on request. This avoids constant connecting and disconnecting from the host application when multiple Web service requests are being serviced.

Figure 7. Connection pool



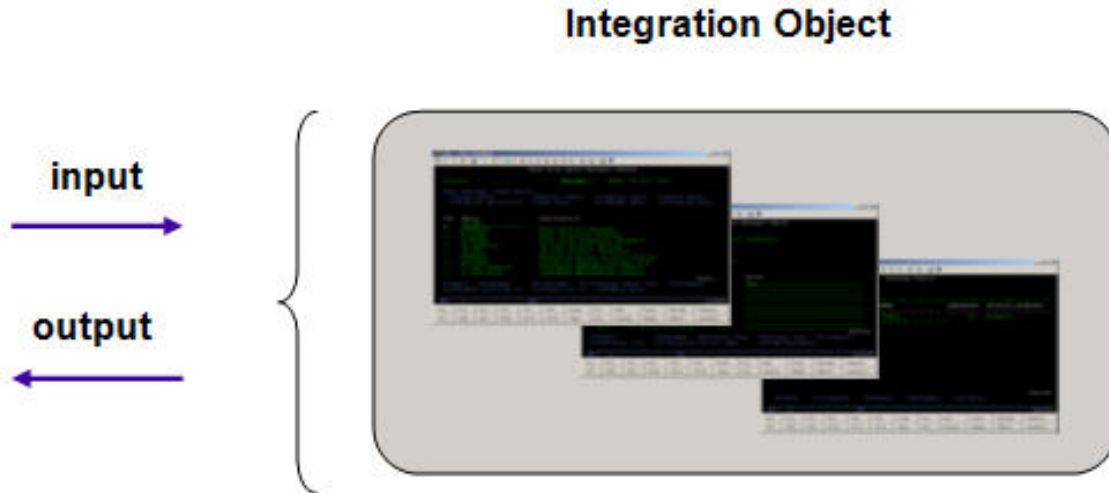
Create Integration Object

A HATS Integration Object is a JavaBean that encapsulates a programmed interaction with a host application. In other words, an Integration Object encapsulates and provides a programming interface to execute a macro.

The Data macro created above is set up to prompt for input, navigate through the application to extract data based on the requested input, and then navigate back to where it can prompt for the next input.

Creating an Integration Object for the Data macro creates a Java programming interface to execute the macro that accepts the input expected by the macro, drives the macro, and supplies as output the output supplied by the macro.

Figure 8. Integration Object

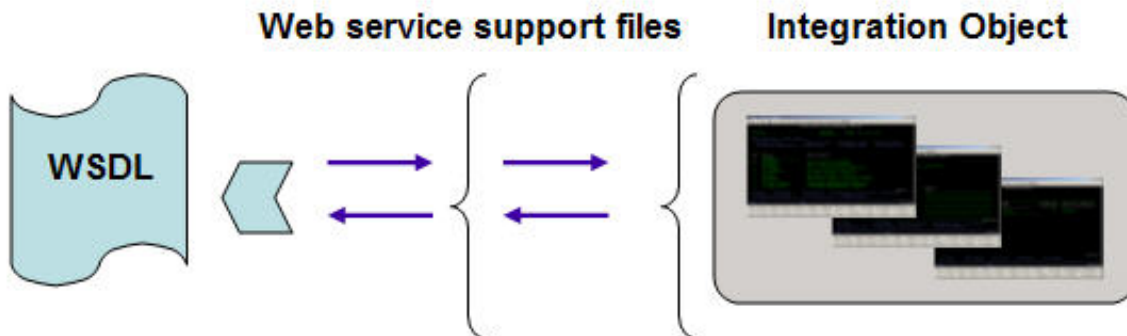


There are many ways Integration Objects can be used to integrate a host application with a new Java or Web-based program. One way is to provide the interaction with a host application for a Web service. One or more Integration Objects can be used by a single Web service.

Create Web service

Once an Integration Object has been created, HATS can be instructed to create Web service support files for it. These are Java class files that describe the methods contained in the Integration Object and its inputs and outputs. The files are also referred to as the wrapper class for the Web service. The inputs and outputs correspond to the inputs and outputs of the Data macro. From the Web service support files, the SDP can create a Web Services Description Language (WSDL) file that describes the interface to the HATS Web service.

Figure 9. Web service



The WSDL file describes how to use the Web service for any client that wishes to do

so. Built in to the SDP is a Web Services Explorer. Using the WSDL file, the Web Services Explorer can be used to test the HATS Web service.

Create and test Web service client

The SDP can also be used to create a sample client application that can call the HATS Web service. The client is built based on the information in the WSDL file that represents the HATS Web service.

After being generated, the sample client application and the HATS Web service application can be tested using the WebSphere test environment built into the SDP.

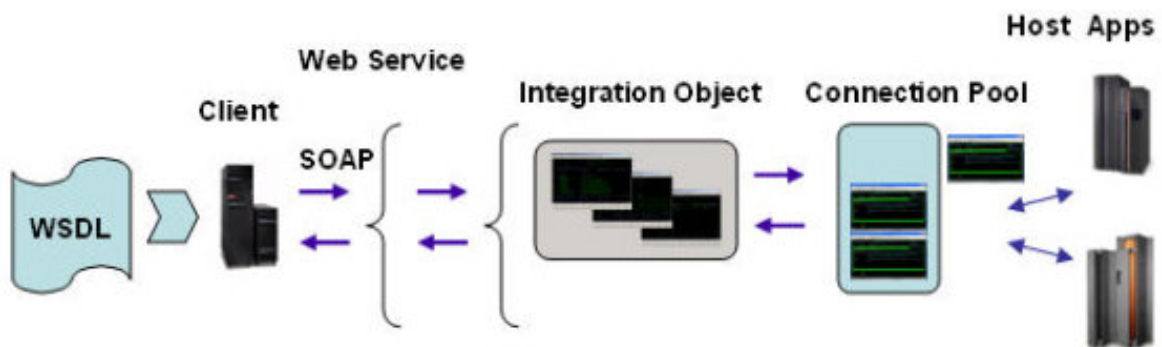
When the WebSphere test environment starts, the HATS application providing the Web service is started. At this point the initial set of connections to the host application for the connection pool is started. The Connect macro is run for each connection leaving each one at the point where the Data macro can begin.

The client application is started and using information from the WSDL file allows entry of an input that corresponds to the input expected by the HATS Web service. This is the same input expected by the HATS Integration Object and the Data macro.

The proxy code in the client application uses a SOAP request to call the HATS Web service with the supplied input. At this point the HATS Web service application instantiates the Integration Object. A free connection from the connection pool is allocated, and the Data macro is driven to navigate through the host application using the input supplied by the client.

As the Integration Object runs, the Data macro collects its output data. This data is then returned in a SOAP response from the HATS Web service to the requesting client.

Figure 10. Web service client



The sample client application can now be used as a base for integrating the host

application with new Java and Web-based applications being developed in the company.

Conclusion

Host applications continue to be a significant part of the mix of applications within an enterprise application suite. Now Web services technology can serve as a common denominator over which applications of different technologies can communicate and be integrated together. Rational HATS can be used non-invasively to provide Web services interfaces for integrating 5250 IBM i and 3270 System z applications with other applications in the enterprise.

Resources

Learn

- Visit the [Rational HATS home page on developerWorks](#) for HATS product information, articles, downloads, demos, and more.
- See [Enterprise Modernization](#) for more about all of the IBM Enterprise Modernization solutions.
- Check out [IBM Enterprise Modernization Sandbox](#) where you can evaluate the IBM Enterprise Modernization solutions (including HATS) for System z and IBM i through practical hands-on experience.
- Visit the [Rational software area on developerWorks](#) for technical resources and best practices for other Rational Software Delivery Platform products.
- Subscribe to the [Rational Edge newsletter](#) for articles on the concepts behind effective software development.
- Subscribe to the [developerWorks weekly newsletter](#) for a weekly update on the best of developerWorks tutorials, articles, downloads, community activities, webcasts and events.
- Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

- Download a trial version of the [IBM Rational HATS Toolkit](#).
- Download other [IBM product trial versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- Using Microsoft® Outlook, Outlook Express, or compatible newsgroup reader, you can view and post to the Rational HATS Newsgroup at: <news://news.software.ibm.com/ibm.software.websphere.hats>.
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

Rick Hardison

Rick Hardison joined IBM in Raleigh, North Carolina in 1970 as a junior programmer. In 1977, he joined an installation support department and provided customer education and installation support for the first releases of IBM networking software. In

1981, he became a product planner and was involved with the announcement of VTAM support for APPN. In 1992, Rick moved into technical marketing providing support for IBM's Communications Clients and Servers, Host On-Demand and Host Access Transformation Services. Rick retired from IBM in 2005 and has since provided consulting services, including technical writing, through Systems Documentation, Inc. (SDI).