Information Management for z/OS

World Wide Web Interface Guide

Version 7.1

SC31-8757-00

Information Management for z/OS

World Wide Web Interface Guide

Version 7.1

SC31-8757-00

Tivoli Information Management for z/OS Web Interface Guide

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, CICS, CICS/ESA, DATABASE 2, DB2, DFSMS/MVS, IBMLink, Language Environment, MVS, MVS/ESA, NetView, OS/2, OS/2 WARP, OS/390, RACF, Redbooks, RMF, SAA, System/390, Tivoli Enterprise Console, TME 10, VTAM, z/OS.

Domino and Lotus are trademarks of Lotus Development Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names mentioned in this document may be trademarks or service marks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Tivoli Information Management for z/OS.

Contents

Preface

This guide describes how you can access Tivoli® Information Management for z/OS from the World Wide Web. Using the World Wide Web, you can use a browser to search for information in Tivoli Information Management for z/OS, and you can also write help desk applications using HTML tags. You can design several views of your system to provide unique universal resource locators to depict various aspects of your system, such as configuration information, change information, and problem information.

There may be references in this publication to versions of Tivoli Information Management for z/OS's predecessor products. For example:

- TME 10™ Information/Management Version 1.1
- Information/Management Version 6.3, Version 6.2, Version 6.1
- Tivoli Service Desk for OS/390® Version 1.2

Who Should Read This Guide

This guide is intended for system and application programmers who will access Tivoli Information Management for z/OS using the Web.

You must be familiar with the information in the *Tivoli Information Management for z/OS Application Program Interface Guide* and the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* before you use this guide.

You must also be familiar with the information in the publications for your operating system, communication protocol, and security product.

Prerequisite and Related Documentation

The library for Tivoli Information Management for z/OS Version 7.1 consists of these publications. For a description of each, see "The Tivoli Information Management for z/OS Library" on page 99.

Tivoli Information Management for z/OS Application Program Interface Guide, SC31-8737-00

Tivoli Information Management for z/OS Client Installation and User's Guide, SC31-8738-00

Tivoli Information Management for z/OS Data Reporting User's Guide, SC31-8739-00

Tivoli Information Management for z/OS Desktop User's Guide, SC31-8740-00

Tivoli Information Management for z/OS Diagnosis Guide, GC31-8741-00

Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications, SC31-8744-00

Tivoli Information Management for z/OS Integration Facility Guide, SC31-8745-00

Tivoli Information Management for z/OS Licensed Program Specification, GC31-8746-00

Tivoli Information Management for z/OS Master Index, Glossary, and Bibliography, SC31-8747-00

Tivoli Information Management for z/OS Messages and Codes, GC31-8748-00

Tivoli Information Management for z/OS Operation and Maintenance Reference, SC31-8749-00

Tivoli Information Management for z/OS Panel Modification Facility Guide, SC31-8750-00

Tivoli Information Management for z/OS Planning and Installation Guide and Reference, GC31-8751-00

Tivoli Information Management for z/OS Problem, Change, and Configuration Management, SC31-8752-00

Tivoli Information Management for z/OS Program Administration Guide and Reference, SC31-8753-00

Tivoli Information Management for z/OS Reference Summary, SC31-8754-00

Tivoli Information Management for z/OS Terminal Simulator Guide and Reference, SC31-8755-00

Tivoli Information Management for z/OS User's Guide , SC31-8756-00

Tivoli Information Management for z/OS World Wide Web Interface Guide, SC31-8757-00

Note: Tivoli is in the process of changing product names. Products referenced in this manual may still be available under their old names (for example, TME 10 Enterprise Console instead of Tivoli Enterprise Console®).

What This Guide Contains

There are now three different platforms through which Tivoli Information Management for z/OS data can be accessed via the Web. :

- "REXX Web connector for MVS -- Overview" on page 3 begins the section on the REXX Web connector for MVS™. Subsequent chapters describe aspects of using the REXX Web connector for MVS.
- "The Database Gateway Application" on page 41 contains information on the Database Gateway Application, with information that is common to both the REXX Web connector for MVS and the REXX Web connector for OS/390.
- "Using Java and JavaScript to Validate Data Fields" on page 53 provides information on how you can use Java™ to validate data fields on the client.
- "REXX Web connector for OS/390 -- Overview" on page 65 describes the REXX Web connector for OS/390.
- "REXX Web connector for OS/2 -- Overview" on page 73 begins the section on REXX Web connector for OS/2®, and subsequent chapters describe the use of this platform.

Typeface Conventions

This guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

Bold Entries that you must use literally, choices, or options that you select appear in **bold**.

Italics Variables and values that you must provide appear in *italics*. New terms also appear in italics.

Monospace
Code examples appear in `monospace` font.

The panels as presented in this guide are not meant to be exact replicas of the way a panel might appear on the screen. The information on the panels is correct, but the spacing is not always exact.

Commands, such as END, CONTROL, RESUME, or DOWN, appear in all capital letters in text. Although not commands, the user responses YES and NO also appear in capital letters.

Contacting Customer Support

For support inside the United States, for this or any other Tivoli product, contact Tivoli Customer Support in one of the following ways:

- Send e-mail to **support@tivoli.com**
- Call 1-800-TIVOLI8
- Navigate our Web site at **http://www.support.tivoli.com**

For support outside the United States, refer to your Customer Support Handbook for phone numbers in your country. The Customer Support Handbook is available online at **http://www.support.tivoli.com**.

When you contact Tivoli Customer Support, be prepared to provide identification information for your company so that support personnel can assist you more readily.

The latest downloads and fixes can be obtained at **http://www.tivoli.com/infoman**

1

Tivoli Information Management for z/OS and the World Wide Web

Overview

The ability to access Tivoli Information Management for z/OS data from the World Wide Web was introduced in Version 6.2.1. The Web connector ran as a started task on an MVS system, and allowed client Web browsers to access MVS Tivoli Information Management for z/OS databases through the Web connector. This version is called the REXX Web connector for MVS. Information about this means of accessing the Web begins on page 3.

In Information/Management Version 1.1, two new alternatives to the REXX Web connector for MVS were provided:

- Tivoli Information Management for z/OS REXX Web connector for OS/390. Information about this means of accessing the Web begins on page 65.
- Tivoli Information Management for z/OS REXX Web connector for OS/2. Information about this means of accessing the Web begins on page 73.

Each of these provides function similar to that of the REXX Web connector for MVS. And in addition to the functions provided by the REXX Web connector for MVS, each of these provides the following functions:

- Multithreading
- Advanced security features

In addition to these functions, Tivoli Information Management for z/OS REXX Web connector for OS/2 provides support for both the TCP/IP and the APPC/MVS communication protocols.

2

REXX Web connector for MVS -- Overview

The REXX Web connector for MVS enables you to access a Tivoli Information Management for z/OS database using a Web browser as a client.

The REXX Web connector for MVS consists of a Web server that implements the Internet Engineering Task Force (IETF) standard protocol HTTP 1.0 and a Database Gateway Application (DGA) that provides Tivoli Information Management for z/OS access functions. The DGA consists of REXX programs and HTML files which are provided as templates that should be modified to suit the requirements of your installation.

The design of the REXX Web connector for MVS assumes there are multiple client machines communicating asynchronously with the REXX Web connector for MVS server. The client and server machines are part of the same IP network and communicate using TCP/IP protocol. The network could be either the Internet itself or a private network (intranet) that has no external connections or is connected to the Internet through a firewall.

Transactions are received by TCP/IP and queued for processing by the REXX Web connector for MVS server.

The REXX Web connector for MVS server interprets requests from client machines, retrieves data from Tivoli Information Management for z/OS using the DGA, and returns response data to the client machine in the form of plain text, Hypertext Markup Language (HTML), or in certain cases, special types of response code such as image formats.

The MVS DGA template that is provided demonstrates one possible use of the REXX Web connector for MVS. Some other possible uses are:

- A helpdesk application which enables users to open problem tickets and check the status of problem tickets they created.
- A retrieval system which enables users to display records stored in Tivoli Information Management for z/OS.
- A front-end application causing one query to be sent to multiple Tivoli Information Management for z/OS systems and combining outputs into a single result list presented to the client.

Access to multimedia information is achieved by retrieving MVS data sets which include multimedia format or by including hypertext references to other servers, usually running in a workstation environment that supports industry standard or proprietary multimedia data formats. Figure 1 on page 4 gives an overall view of the REXX Web connector for MVS.

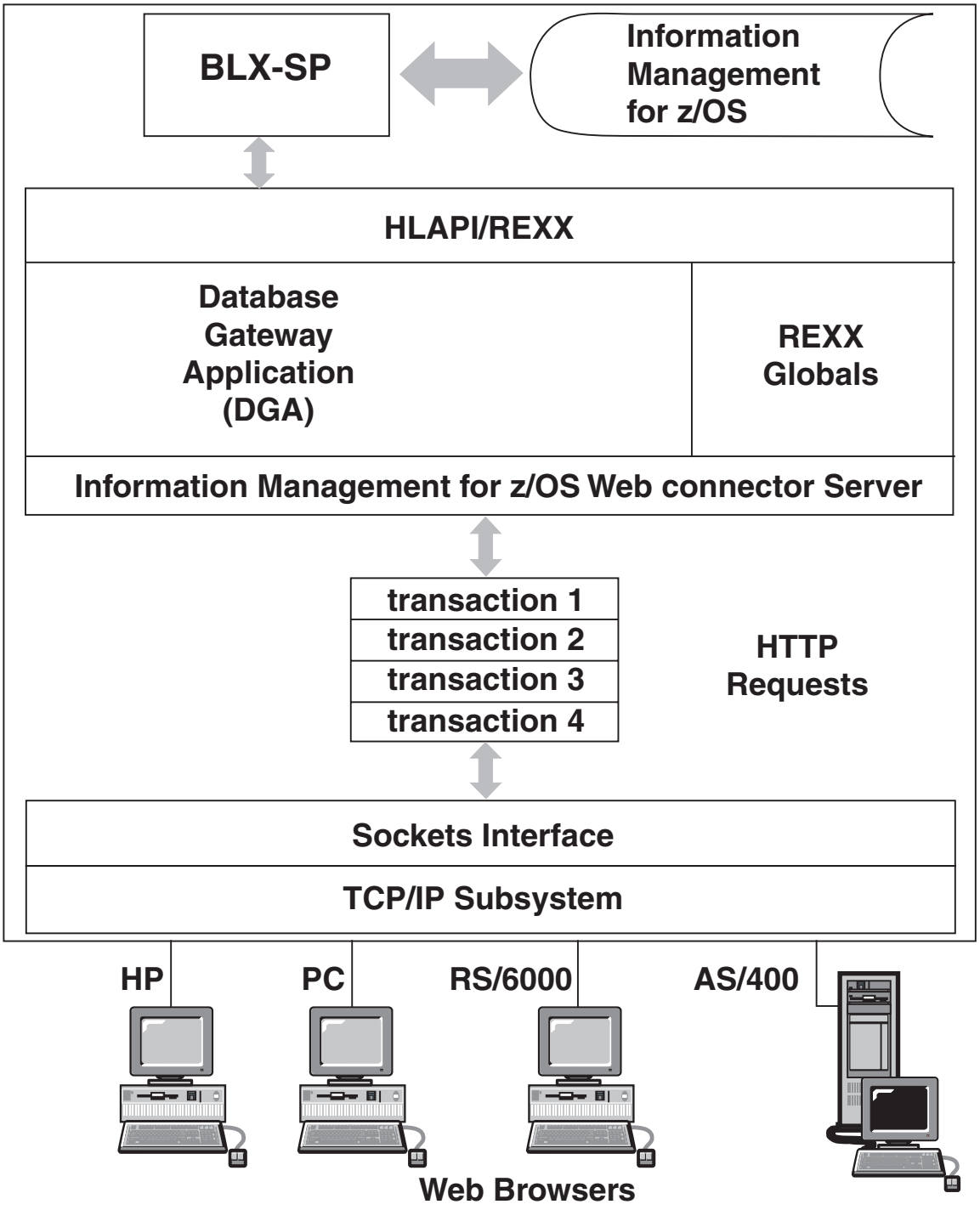


Figure 1. Overview of the REXX Web connector for MVS Components

Major components can be divided into *Client* components and MVS/ESA™ *Host* components (the Tivoli Information Management for z/OS database).

Client Components

The client should be a TCP-capable computer (either a workstation or a mainframe) with Web browser software. Most commercially available browsers which support forms processing should work when connecting to the REXX Web connector for MVS.

You need to do only a small amount of customizing in the network components of this application. As you will see in "Loading the REXX Web connector for MVS Home Page from a Client Browser" on page 14, an existing Web client can begin using the REXX Web connector for MVS application simply by loading the appropriate Uniform Resource Locator (URL).

MVS/ESA Overview

HTTP messages are forwarded into the REXX Web connector for MVS server from the network through the MVS TCP/IP subsystem. The REXX Web connector for MVS server parses these requests. Requests for static HTML or binary data code are served immediately by reading the information either from the HTML data set or from an internal cache. Requests for HTML forms are passed to the Database Gateway Application (DGA) for processing. The Database Gateway Application utilizes High Level Application Program Interface/REXX (HLAPI/REXX) calls to communicate with Tivoli Information Management for z/OS. The DGA performs the required operations, formats the data received from Tivoli Information Management for z/OS into HTML code or plain text, and passes it back to the REXX Web connector for MVS server to be returned to the client.

The following sections discuss three MVS/ESA components:

- The Tivoli Information Management for z/OS program product
- The REXX Web connector for MVS server
- The Database Gateway Application

Tivoli Information Management for z/OS Considerations

From the standpoint of Tivoli Information Management for z/OS, the Database Gateway Application is a program which uses the standard HLAPI/REXX interface to access the Tivoli Information Management for z/OS database. Any feature or function available through the HLAPI/REXX interface may be used in the Database Gateway Application. The *Application Program Interface Guide and Reference* (SC34-4463) provides detailed information on the HLAPI/REXX interface.

The REXX Web connector for MVS Web Server

The REXX Web connector for MVS is a specialized Web server which receives client HTTP 1.0 requests, retrieves data from Tivoli Information Management for z/OS using the Database Gateway Application, and sends a response. The REXX Web connector for MVS can interpret either HTML or plain text, and can retrieve static HTML files as well as multimedia image format files on MVS.

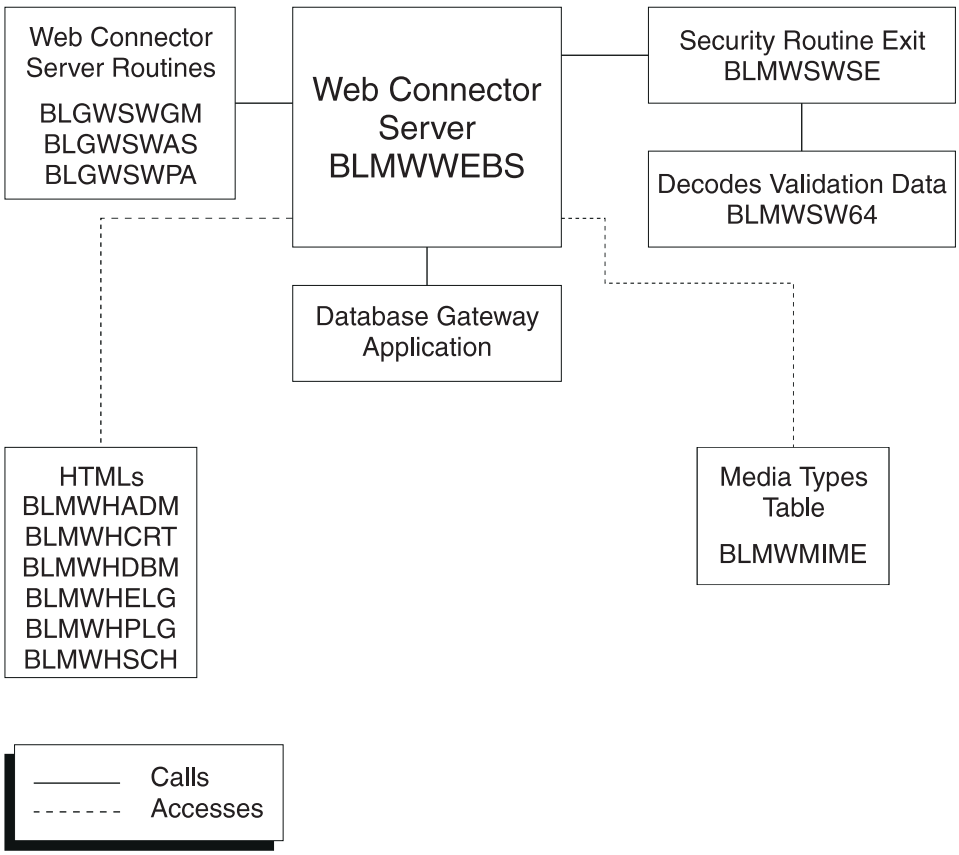


Figure 2. The REXX Web connector for MVS Web Server

The Database Gateway Application

The Database Gateway Application contains program logic designed for using the Tivoli Information Management for z/OS HLAPI/REXX interface and for generating dynamic HTML either from Tivoli Information Management for z/OS records or from search results lists. The sample DGA provided demonstrates the basic techniques needed to communicate with Tivoli Information Management for z/OS, and also demonstrates a way of interacting with a Web client. The Database Gateway Application is described in greater detail in “Modifying the Database Gateway Application” on page 47.

3

REXX Web connector for MVS -- Installation and Operations

Prerequisites

The prerequisites for running the REXX Web connector for MVS are:

- A Web browser
- IBM® Library for SAA® REXX/370 Release 3 for MVS/ESA (5695–014)

Note: You must use the IBM TCP/IP product (which is included with OS/390) because it supports REXX Sockets. Other vendors' TCP/IP products are only supported if they include REXX Sockets.

Running the REXX Web connector for MVS Server as an MVS Batch Job

The REXX Web connector for MVS server runs as a TSO command under control of the TSO command processor program IKJEFT01. The name of this command is BLMWWEBS. Incoming requests from client Web browsers are queued by TCP/IP and passed to BLMWWEBS one at a time.

Sample JCL is provided in SBLMSAMP member BLMWJCL, shown in Figure 3 on page 9. When you are testing the REXX Web connector for MVS, you should submit the REXX Web connector for MVS as a batch job. After you have tested it, you may want to make it a started task, described in "Running the REXX Web connector for MVS as an MVS Started Task" on page 15. To run it as a batch job, follow these steps:

1. Allocate a partitioned data set (PDS) to hold any REXX routines you will copy, modify, or create. The PDS that you allocate should have the same data set organization as SBLMREXX. At a minimum, allocate the equivalent of 10 tracks on a 3390. This user PDS should be concatenated ahead of the SBLMREXX data set pointed to by the SYSEXEC DD statement in the sample BLMWJCL.
2. Edit the sample JCL provided in BLMWJCL. Read the comments and modify the data set names to match your installation. Change the JOB card as needed. Ensure that the STEPLIB concatenation contains any additional loadlibs, such as session members, that may be necessary. Review the parameters on the BLMWWEBS commands and change the TCPIP(), TASKID(), OWNER(), and HTML() parameters to match the values of your installation. Because the sample application uses Tivoli-supplied PIDTs, be sure that either your session member points to the Tivoli-supplied SBLMFMT data set or use a DD card in the JCL (typically RFTDD) for SBLMFMT.

3. Copy members BLMWSFIN and BLMWSWSE from the SBLMREXX data set to the user PDS created in the first step of this procedure.

   Note: Do not modify the sample application members in SBLMREXX. Always copy them to your REXX PDS and make any needed changes there.

4. Edit the BLMWSFIN member in your REXX PDS and change the APPLICATION_ID, SESSION_MEMBER, and PRIVILEGE_CLASS as needed for your test setup. The *Tivoli Information Management for z/OS Application Program Interface Guide* contains additional information about control parameter data blocks (PDBs) used by the HLAPI/REXX interface. If you would like the Webmaster (the user specified on the OWNER() parameter of BLMWWEBS) to be able to issue Server commands (described in "REXX Web connector for MVS -- Commands" on page 21), then you will need to edit the BLMWSWSE member in your REXX PDS; you will need to change RETURN 8 USERID; to RETURN 0 USERID; at the end of the BLMWSWSE member. This change will allow all Web browsers to connect to the REXX Web connector for MVS Server, but only the Webmaster will be able to issue commands. You should make this change, and also specify AUTHORITY(NO) on the BLMWWEBS command while you are becoming familiar with REXX Web connector for MVS.

   Note: BLMWJCL is shipped with AUTHORITY(NO). You should carefully review "REXX Web connector for MVS -- Security Considerations" on page 17 before changing this parameter.

5. Ensure that TCP/IP is active
6. Ensure that the IBM Library for SAA REXX/370 is accessible. It must be in LPA, LINKLIST, or the JCL STEPLIB.
7. Ensure that the Tivoli Information Management for z/OS BLX Service Provider (BLX-SP) is active
8. Submit the BLMWJCL job stream to start the REXX Web connector for MVS
9. Go to "Test from a Client" on page 15

Figure 3 on page 9 shows the BLMWJCL provided in SBLMSAMP:

```
//WEBDEV  JOB '     ',TIME=1440,                                          00010000
//           REGION=8M,                                                   00020000
//           CLASS=A,MSGCLASS=a,MSGLEVEL=(1,1)                            00030000
//*--------------------------------------------------------------------* 00140000
//* INFOWEB Server JCL                                 -                 00150000
//*                                                    -                 00160000
//*                                                    -                 00170000
//* DDNAME    Description                              -                 00180000
//* ========  =======================================  -                 00190000
//* STEPLIB   BLM.SBLMMOD1                             -                 00200000
//* SYSEXEC   Rexx procedure library                   -                 00210000
//*           - your.user.rexx (optional)              -                 00220000
//*           - BLM.SBLMREXX                           -                 00230000
//* RFTDD     Change the following DD to point to   @P1A-                00240000
//*           your customized RFT if you have one   @P1A-                00250000
//*           Always concatenate the Tivoli supplied@P1A-                00260000
//*           SBLMFMT dataset                          @P1A-             00270000
//*           - your.api.pidts                         @P1A-             00280000
//*           - BLM.SBLMFMT                       @P1A@P2C-              00290000
//* HLAPILOG  (optional) For HLAPI/Rexx debugging       -                00300000
//* APIPRT    (optional) For HLAPI/Rexx debugging       -                00310000
//* SYSTSPRT  For TSO terminal monitor program          -                00320000
//* SYSOUT    For TSO terminal monitor program          -                00330000
//* SYSPRINT  For TSO terminal monitor program          -                00340000
//*                                                    -                 00350000
//* SYSTSIN   TSO terminal monitor program input       -                 00360000
//*                                                    -                 00370000
//* Comments:                                          -                 00380000
//*          Update the jobcard and data set names     -                 00390000
//*          where appropriate.                        -                 00400000
//*          Update the BLM.SBLMxxxx data set names    -                 00410000
//*          to the names used at your installation.   -                 00420000
//*          BLMWWEBS lower case parameters should     -                 00430000
//*          be substituted with your values.          -                 00440000
//*                                                    -                 00450000
//* CHANGE ACTIVITY                                    -                 00460000
//* $L0= P4046   JOY6212 960215 DMGPGTR: INFO/MGMT 6.2.1-                00470000
//*                                      Web connector -                 00480000
//* $P1= P4226   JOY6301 960805 DMGPGTR: Added RFTDD   -                 00490000
//*                                      statement     -                 00500000
//* $P2= P4241   JOY6301 960815 DMGPPJL: fix typo      -                 00510000
//* $01= OW24937 JOY6301 970731 DMGPLDM: Fix for Huge  -                 00520000
//*                                      sends         -                 00530000
//* $02= OW26576 JOY6301 970731 DMGPLDM: Stop reposting -                00540000
//*                                      Netscape forms -                00550000
//* $L1= DCR521  HOYT100 970721 DMGPGTR: OS/390 web    -                 00560000
//*                                      connector     -                 00570000
//*-----------------------------------------------------                 00580000
```

Figure 3. Sample JCL provided in BLMWJCL, a member of SBLMSAMP. Part 1 of 2

```
//STEP0100 EXEC PGM=IKJEFT01,PARM='PROFILE NOPREFIX NOWTPMSG'          00590000
//STEPLIB  DD DISP=SHR,DSN=BLM.SBLMMOD1                               00600000
//SYSEXEC  DD DISP=SHR,DSN=your.user.rexx                             00610000
//         DD DISP=SHR,DSN=BLM.SBLMREXX                               00620000
//RFTDD    DD DISP=SHR,DSN=your.api.pidts                             00630000
//         DD DISP=SHR,DSN=BLM.SBLMFMT                                00640000
//HLAPILOG DD SYSOUT=*,OUTLIM=20000                                   00650000
//APIPRINT DD SYSOUT=*,OUTLIM=20000                                   00660000
//SYSTSPRT DD SYSOUT=*,OUTLIM=20000                                   00670000
//SYSPRINT DD SYSOUT=*,OUTLIM=20000                                   00680000
//SYSOUT   DD SYSOUT=*,OUTLIM=20000                                   00690000
//SYSTSIN  DD *                                                       00700000
%BLMWWEBS +                                                           00710000
  TCPIP(TCPIP) +                                                      00720000
  PORT(8000) +                                                        00730000
  BACKLOG(256) +                                                      00740000
  TASKID(ibmuser) +                                                   00750000
  OWNER(ibmuser)  +                                                   00760000
| HTML('BLM.VxRxMx.SBLMHTMV') +                                       00770000
  TRACE(YES) +                                                        00780000
  EXECFLOW(YES) +                                                     00790000
  CMDPREFIX('!') +                                                    00800000
  RECVTIMEOUT(30) +                                                   00810000
  TIMEZONE(EST) +                                                     00820000
  CACHESIZE(10) +                                                     00830000
  AUTHORITY(NO) +                                                     00840000
  DEBUG(NO) +                                                         00850000
  SEGMENTSIZE(16384)+                                                 00860000
  SENDTIMEOUT(120)+                                                   00870000
  PRAGMA(CACHE) +                                                     00880000
  LIFESPAN(30)  +                                                     00890000
  REALM('INFOWCF') +                                                  00900000
  MEDIATYPETABLE('BLM.V1R2M0.SBLMSAMP(BLMWMIME)') +                   00910000
  DOCUMENTROOT('ibmuser')                                             00920000
/*                                                                    00930000
```

Figure 4. Sample JCL provided in BLMWJCL, a member of SBLMSAMP. Part 2 of 2

BLMWWEBS Parameters

These are the parameters that you can specify in the invocation of BLMWWEBS. Operands separated by an OR symbol (|) mean you choose one of them. If you omit a parameter or specify it as a blank, the default value is invoked.

TCPIP(name)

The name of the TCP/IP started task in use at the system where the REXX Web connector for MVS is running. The default value is TCPIP.

PORT(number)

The port number to be used by the REXX Web connector for MVS. If another Web server is running in the same MVS system, the port number must be changed to avoid conflict. You should check the port number with your TCP/IP coordinator to ensure that it does not conflict with registered port numbers or port numbers already in use in your system. Valid port numbers are in the range 1-65534. If you specify a value outside of the 1-65534 range, then a port number of 80 is assigned. The default value is 80.

If you would like to reserve a port number for use by the REXX Web connector for MVS, have your TCP/IP coordinator define a PORT statement in the TCP/IP configuration file.

Note: If you choose to reserve a port number for the REXX Web connector for MVS (running in batch or as a started task), be sure that the port number in the TCP/IP configuration file matches the PORT() parameter used by the REXX Web connector for MVS. See *TCP/IP for MVS V3R1 Customization and Administration Guide* for more information.

BACKLOG(number)

The backlog queue value used by the TCP/IP address space when a connect request arrives to the REXX Web connector for MVS server and the server is already connected to another client and is busy serving that client's request. TCP/IP will use this value as the maximum number of connection requests that can be queued. If further connection requests arrive, they will be rejected by TCP/IP. The client Web browser will indicate this with a *connection refused* message. The maximum value for this parameter is 1024. The default value for this parameter is 256.

TASKID(name)

This should be 1 to 8 characters used as a unique name for this server as its task identification to the TCP/IP system. The default is the TSO userid assigned to the REXX Web connector for MVS.

OWNER(name)

The Webmaster for this server. This user will be able to issue commands to the REXX Web connector for MVS server. BLMWWEBS compares the value specified in the OWNER() parameter to the value entered by the users at their browsers when they are prompted for authorization. If there is a match, commands will be accepted. The default value for this parameter is WEBMASTER.

HTML(pds-name)

The name of the partitioned data set (PDS) that contains HTML files to be used by REXX Web connector for MVS. It defaults to <userid>.HTML, where <*userid*> is the TSO userid assigned to the REXX Web connector for MVS.

TRACE(YES|NO)

This flag indicates to the server whether to issue detail messages related to activity the server is performing. This flag can also be changed by REXX Web connector for MVS commands (see "REXX Web connector for MVS Server Commands" on page 21). The default is NO.

EXECFLOW(YES|NO)

This flag indicates to the server whether to issue message number 024 indicating the server routine name and the parameters used by the routine. This flag can also be changed by REXX Web connector for MVS commands (see "REXX Web connector for MVS Server Commands" on page 21). The default is NO.

CMDPREFIX(string)

This character string indicates to the server whether to treat the message as a command. Quotes or apostrophes cannot be part of this string because they are used as delimiters. The default is the exclamation mark **!**.

RECVTIMEOUT(number)

The duration in seconds to wait for data available at the TCP/IP socket coming from a client. This wait time applies when a connection has already been established and some data has been received from the client, but it is not yet a complete HTTP request. The maximum value for this parameter is 300. The default value for this parameter is 15.

TIMEZONE(hh:mm |EST|CST|MST|PST|ALA|HAW)

The number of hours and minutes of displacement from the GMT (Greenwich Mean Time) zone. This value is used to calculate the date and time values sent back to the client as part of the HTTP response. For example, −5:00 or −5 is the US EST time zone, which is GMT minus 5 hours. Displacement values for hours and minutes must be in the HH:MM format. Hours alone are accepted, as are the values of standard US time zones. If this value is omitted or invalid, the default is 00:00. Remember to adjust these values for daylight savings time.

CACHESIZE(number)

The size in megabytes of the REXX Web connector for MVS server *static* cache, which is usually for media data of type HTML, HTM, and TXT. Refer to “The Media Types Table” on page 33 for more information on caching. *Dynamic* data is never cached by the REXX Web connector for MVS server, and contains HTTP headers that prevent proxy servers and Web browsers from caching it. The maximum value for this parameter is 50. The default value for this parameter is 8. Specifying CACHESIZE(0) will stop the caching feature. This may be useful when developing HTML pages to prevent an out-of-date version of the page from being shown.

AUTHORITY(YES|NO)

Specifies whether you want Tivoli Information Management for z/OS REXX Web connector for MVS security routines enabled to activate the use of the HTTP Basic authentication scheme. More information on the authentication can be found in “REXX Web connector for MVS User Authentication” on page 18. The initial setting of this field as shipped in the BLMWJCL is NO. A setting of AUTHORITY(YES) will cause your browser to display an authority window on which the user must enter an Authentication string the first time the REXX Web connector for MVS is accessed. The Authentication string is verified by the security exit BLMWSWSE. If the string is verified as correct, then the current and subsequent browser transactions are processed. The default value for this parameter is YES.

DEBUG(YES|NO)

This flag indicates to the server that additional output useful for debugging problems in BLMWWEBS should be created. This additional output is useful when working with the Tivoli Services group. The DEBUG command can also be used to turn the DEBUG option on and off. The default value for this parameter is NO.

SEGMENTSIZE(number)

The size in bytes of each segment sent by the server on each socket send. Valid specifications for this parameter are in the range 2048–64536. It is recommended that SEGMENTSIZE be equal to one-half or less than the value used by the TCP/IP parameter DATABUFFERPOOLSIZE. SEGMENTSIZE must not exceed DATABUFFERPOOLSIZE or the sends will fail during times of high TCP/IP traffic or data transfers. The default value for this parameter is 4096.

SENDTIMEOUT(number)

The number of seconds the server will wait for a send buffer to become available. Valid values for this parameter are in the range 30–64536. The default value for this parameter is 120 seconds.

LIFESPAN(nnnn|FOREVER)

The number of minutes that a dynamically created form will be considered valid or not expired. Valid values are 0 to 1440 or the word FOREVER. If 0 is specified, then all forms will be sent expired. If FOREVER is specified, then all forms sent will not expire. If a number in the range of 1–1440 is used, all forms sent will

expire after that number of minutes has passed. See “InfoWeb Directive Support - Expiring a Document” on page 33 for a method to expire individual forms. The default value for this parameter is 30 minutes if LIFESPAN is omitted or an invalid value is specified.

PRAGMA(CACHE|NOCACHE)

This parameter controls the ability of a proxy server or gateway to cache dynamic HTML that is sent to the client browser by the Web connector feature server. Specifying PRAGMA(CACHE) allows the proxy server or gateway to cache dynamic HTML, and is the default if PRAGMA is omitted or an invalid value is specified. Specifying PRAGMA(NOCACHE) causes the Web connector server to add PRAGMA(NOCACHE) to the HTTP header, which prevents a proxy server or gateway from caching dynamic HTML. Specifying PRAGMA(CACHE) is recommended.

DOCUMENTROOT(prefixvalue)

This parameter specifies a prefix to be added when algorithmic mapping of a URL to a data set is used. This provides an additional level of security by restricting the data set names that can be referenced by the REXX Web connector for MVS. For example, if the document root is INFOMAN.WEB, then any data set generated by the algorithm will start with the qualifier INFOMAN.WEB. A URL such as http://mvs20:8000/main.html will reference INFOMAN.WEB.HTML(MAIN) (see “Static-URL to Data Set Mapping” on page 31 for details about algorithmic mapping of a URL to a data set. If this value is omitted, the current userid is used. A value specified for DOCUMENTROOT neither overrides nor depends on the coding of the HTML() parameter.

REALM(name)

This parameter provides a means to customize the signon prompt used by the Web browser when AUTHORITY is set to YES (see the description of the AUTHORITY parameter of the BLMWWEBS command). The string specified as REALM will be inserted in the signon prompt. Different Web browsers use different formats for this signon prompt, so you should verify that the signon prompt issues a meaningful message by performing some testing in your operating environment. The value of the REALM parameter can be temporarily overridden by the security verification routine, BLMWSWSE. If the format of the string returned by BLMWSWSE is a return code 1 followed by an informational message, the message will substitute the REALM for the transaction in process. This allows for better communication from the security verification routine; for example, if the error message indicates that the password entered has expired and a new password should be entered, this detail can be clearly indicated by using this message. If this parameter is omitted, the value *INFOWCF* is used for the REALM name.

MEDIATYPETABLE(data set)

This parameter is the data set name of the file that contains the mime types table. Please refer to “The Media Types Table” on page 33 for a description of this table. The initial setting of this parameter is BLM.VxRxMx.SBLMSAMP(BLMWMIME). If the parameter is missing, the default is <userid>.media.

Loading the REXX Web connector for MVS Home Page from a Client Browser

To load the sample REXX Web connector for MVS home page in your Web browser, you must know the IP host name for the MVS host and the TCP/IP port number. (The TCP/IP port number was coded in the PORT() parameter described in "BLMWWEBS Parameters" on page 10). The REXX Web connector for MVS server log will show you the IP address and host name for your MVS/ESA system and the TCP/IP port number; see "REXX Web connector for MVS -- Logging" on page 23 for an example of the REXX Web connector for MVS server log.

"Dynamic-URL Mapping to a Forms Processing Routine" on page 30 shows a sample URL structure. An example of the URL to load the REXX Web connector for MVS home page might be:

```
http://9.152.64.23:8000/infoweb/blmwhdbm.html
```

where

- `http://` is the header required by protocol
- 9.152.64.23 is an example IP address. Replace this with the IP address for your MVS host (if a hostname for the IP address is defined in the network, it can be used instead; for example, MVS01 might then be substituted for 9.152.64.23).
- `:8000` is the port address. Use the value from the PORT() parameter of the BLMWWEBS command.
- `/infoweb/` is a marker used to simplify URL parsing on the REXX Web connector for MVS
- `blmwhdbm.html` is the home page name for the sample application.

Once you develop applications for the REXX Web connector for MVS, you may want to use a different home page. The IP and port will not change unless you start another Web connector or change the host IP address.

Stopping the REXX Web connector for MVS

To stop the REXX Web connector for MVS, on the Web browser enter the !CLOSE command (our examples use the default character ! as the command prefix; you may choose to specify a CMDPREFIX() with a different character).

```
http://9.67.51.12:8000/!close
```

To enter the !CLOSE command, or any other command, you must be an authenticated user equal to that of the OWNER() parameter in the JCL parameters.

Note: Once you have closed the REXX Web connector for MVS, you may need to wait several minutes before you restart it in order to give TCP/IP time to clear its socket connections.

Using the !CLOSE command from a browser is the preferred method of stopping the REXX Web connector for MVS, whether running in batch or as a started task. The MVS CANCEL command can be used, but depending on the status of the REXX Web connector for MVS at the time the CANCEL is issued, you may receive a system abend of 13E, 222, A03, or D23.

Test from a Client

In order to test the REXX Web connector for MVS, do the following:

1. Ensure that your client machine is connected to the MVS host via an IP network.
2. Ensure that the REXX Web connector for MVS is active on MVS.
3. Start a Web browser on your client machine.
4. Connect to the REXX Web connector for MVS Home Page as described in "Loading the REXX Web connector for MVS Home Page from a Client Browser" on page 14. Once you have loaded the home page, you can navigate through the REXX Web connector for MVS application using hyperlinks.

   Note: An error screen may appear that looks like

   ```
   Sorry ...
   The request from your web client
   GET URI: infoweb/blmwhdbm
   has failed, reason was : File name not found
   HTTP response code: 404 (Not Found)
   From server at
   Running: InfoWCF/1.0
   ```

   If this occurs, look closely at the GET URI line. It should always say `GET URI: INFOWEB/nnnnnn` (if you are not using the DOCUMENTROOT parameter), where `NNNNNN` should be `BLMWHDBM` if you are using the supplied sample application. If INFOWEB is missing, add **INFOWEB/** to the URL and try again. If **INFOWEB/** is already present, then ensure that the data set pointed to by the HTML() parameter in the BLMWWEBS command contains BLMWHDBM.

Running the REXX Web connector for MVS as an MVS Started Task

After you have tested the REXX Web connector for MVS as a batch job, you may want to run it as an MVS started task.

To run the Web connector feature as a started task, you must define a procedure for it in SYS1.PROCLIB or the procedure library in the JES proclib concatenation. These are the steps:

1. Create a member in the procedure library (proclib).
2. Copy the BLMWJCL job stream into the member you created in the proclib and make the following changes to the JCL:
   - Remove the Job card.
   - Remove the /* from the end of the BLMWJCL.
   - Remove the in-stream DD data used for //SYSTSIN. An in-stream data set cannot be used in a proc. Therefore, place the in-stream DD data into a data set. The in-stream data is the BLMWWEBS command and its parameters. The new data set should have LRECL=80 RECFM=FB, and it can be a member of a partitioned data set (PDS). Do not use a data set in the //SYSEXEC concatenation.
   - Change the //SYSTSIN DD to use the DSN= for the data set you created that now contains the BLMWWEBS command and its parameters. Add DISP=SHR. For example,

     ```
     //SYSTSIN   DD DSN=webcon.parms,DISP=SHR
     ```

3. Ensure that TCP/IP is active.
4. Ensure that the Tivoli Information Management for z/OS BLX Service Provider (BLX-SP) is active.
5. Have the MVS system operator issue the MVS start command:

   ```
   S procname
   ```

   where PROCNAME is the name of the member you created in the proclib that contains the procedure to run the REXX Web connector for MVS.

4

REXX Web connector for MVS -- Security Considerations

Two different environments must be considered when discussing security issues:

- The REXX Web connector for MVS operating in an int**ra**net environment only
- The REXX Web connector for MVS operating in the Int**er**net (and possibly in an int**ra**net at the same time)

A "firewall" completely blocking access to the REXX Web connector for MVS is equivalent to operating in an intranet, if the firewall completely shields the REXX Web connector for MVS from the Internet. For the purpose of this discussion, we will stipulate that an int**ra**net is a secure environment, with no access to the external Int**er**net.

The REXX Web connector for MVS Operating in an Intranet

Because the Resource Access Control Facility (RACF®) authority assigned to the REXX Web connector for MVS provides another layer of security, careful consideration should be given to the access authority provided. The REXX Web connector for MVS should be assigned RACF access authority sufficient only to perform the transactions defined in the Database Gateway Application (DGA).

Security Service Routines

All transactions performed by the REXX Web connector for MVS are defined by the DGA. As with any other database application, making the DGA transactions secure is fundamental to the overall security of your installation. The REXX Web connector for MVS user access is controlled by the Security Service Routines:

BLMWSWSE	Web Service Security Exit
BLMWSW64	Web connector Decode String Routine

RACF Privileges and Authorizations

RACF authorization is performed at the time the IKJEFT01 session is started, either as a batch job, as a started task, or as an interactive user.

All transactions run by the DGA share the attributes of the RACF userid that was either specified or assigned by the OS/390 system to the REXX Web connector for MVS server.

This justifies using a very conservative RACF profile for the REXX Web connector for MVS user ID. In the unlikely event that all other barriers have been passed, RACF will restrict any further access within your OS/390 system to a very limited area.

Note: Remember that the DGA does not give full access nor is it a command shell by itself. It only runs transactions defined by the Forms Processing Routines.

RACF privileges or access to data sets, other than the ones allocated by the REXX Web connector for MVS procedure or JCL, are not required.

If you utilize a user ID with special authority for started tasks in your MVS/ESA environment, and you plan to make the REXX Web connector for MVS a started task, you should consider providing it with a different RACF userid, with access authorities and privileges that are the minimum required to permit its operation.

REXX Web connector for MVS User Authentication

"BLMWWEBS Parameters" on page 10 describes the BLMWWEBS parameters that you will modify to control some of the functions of the REXX Web connector for MVS. One of the parameters in BLMWWEBS is AUTHORITY(<YES/NO>). If AUTHORITY(YES) is specified in the invocation of BLMWWEBS, the REXX Web connector for MVS implements what is defined in HTTP protocol as the Basic authentication scheme. No other security protocols are available directly from the REXX Web connector for MVS, although it could be combined with a proxy server that provides additional security protocols, or by other combinations of software and hardware that provide added security.

The Basic authentication scheme enables an encoded userid/password pair to be passed to the REXX Web connector for MVS at the time an HTTP transaction is initiated. The server, BLMWWEBS, calls the security routine BLMWSWSE. BLMWSWSE calls BLMWSW64, which uses an algorithm called Base 64 decoding to decode the information and return it to the server.

Note: The Base 64 algorithm is an Internet standard defined by the IETF Request For Comments (RFC) number 1421.

BLMWSWSE allows or denies access to a client user by returning specific codes and information to the REXX Web connector for MVS server:

- To allow a user to have access, BLMWSWSE should return RC=0, along with a userid that will be passed as a parameter to BLMWSWRT.
- To deny access to a user and return the REALM() value, BLMWSWSE should return RC=8. See the explanation for the REALM parameter in "BLMWWEBS Parameters" on page 10.
- To deny access and return a replacement string to use as the REALM value, BLMWSWSE should return RC=1 along with the string used to replace the REALM() value.
- To allow BLMWSWSE to have BLMWSWRT called, BLMWSWSE should return RC=4 followed by a user id, if desired, and a member name to be used by BLMWSWRT. A comma must be returned to separate the userid and the member name. The member name is required, and if this is omitted or the return data does not contain the comma, then the server will treat this as RC=8 and return the REALM() value to the client. If the returned data is correct, then the server will discard any data received from the client and call BLMWSWRT using the member name as the URI. The user can add additional REXX code to BLMWSWRT to determine what information the client will receive.

In the following examples of return statements, assume that userid is a variable:

- Allow access for userid:

```
return 0 userid;
```

- Deny access and use REALM():

```
return 8;
```

- Deny access and use REALM(); userid is ignored:

```
return 8 userid;
```

- Deny access, use ABC HelpDesk to replace the REALM() value:

```
return 1 'ABC HelpDesk;
```

- BLMWSWRT will be called and to it will be passed userid and the member name EXPIRED:

```
return 4 userid ',EXPIRED';
```

- BLMWSWRT will be called. A null string will be passed as the userid parameter to BLMWSWRT. Member name REVOKED will be used.

```
member='REVOKED'
return 4 ','member;
```

The model BLMWSWSE routine should be customized to implement the security standards of your installation. You should carefully review the different authentication schemes and their advantages and disadvantages; once a scheme is chosen, you can modify BLMWSWSE to implement it.

The REXX Web connector for MVS Operating in the Internet

Internet security is an important issue that must be given significant consideration. Any security scheme that you consider should be carefully reviewed by your installation security staff and perhaps even validated by external consultants.

Most MVS/ESA systems do not connect directly to the Internet, but are protected by firewalls and other security mechanisms.

In general, you should not connect the REXX Web connector for MVS to the Internet without using an intermediate security layer. This security layer should provide another level of userid/password protection, and possibly a data encryption mechanism.

The objective of this security layer is to ensure that:

- Data transmission on the Internet is secure, that is, packet data is protected while it travels the public network.
- The user of the REXX Web connector for MVS is an authorized user of the server, and the access codes provide verification of identity.

Once those two conditions are met, the Internet user should be treated in the same way as an intranet user.

Securing Your Database Gateway Application (DGA)

The DGA can provide an additional layer of security. In order to create a secure DGA, you should follow some rules to prevent unauthorized access to your code.

- Avoid the use of the REXX Interpret statement
- Do not provide the capability of invoking TSO commands
- Use a privilege class for the REXX Web connector for MVS that provides minimum access to Tivoli Information Management for z/OS data.

The REXX Interpret Statement

You should not use the Interpret statement when coding your DGA router or your Forms Processing Routines (FPRs). If you use the Interpret statement, it could make your DGA vulnerable to a Trojan Horse embedded in an HTTP request that could use a REXX statement to overwrite, delete, or access data.

If you do use the Interpret statement in your Database Gateway Application code, you should ensure that any string you are going to interpret does not contain REXX statement separator characters; allowing REXX statement separator characters is a security exposure.

Note: CGI scripts written in Perl or other interpreted languages are subject to the same security exposure. Such an exposure is not unique to REXX Web connector for MVS, but is a consideration for most available Web servers.

TSO Command Invocation

If you permit TSO commands to be run from your Database Gateway Application FPRs, you may provide an opening for a Trojan Horse attack, as variable data entered from a form field may include TSO command delimiters and be used to change the intended FPR function. This security exposure is similar to that described for the Interpret statement.

Access to Data in the Tivoli Information Management for z/OS Database

The Tivoli Information Management for z/OS Application Programming Interface is entirely under the control of its privilege class. The privilege class is controlled by the application_id and privilege_class values coded in BLMWSFIN, described in “DGA REXX Forms Service Routine BLMWSFIN Interface” on page 46 and in the individual forms processing routines beginning with “DGA REXX Forms Processing Routines (FPRs)” on page 42. Specifying appropriate values here will enable you to restrict access of the Web browser to data in the Tivoli Information Management for z/OS database. Using a privilege class with an authority of master is not recommended. Additional information about the privilege class concept can be found in *Tivoli Information Management for z/OS Application Program Interface Guide* and *Tivoli Information Management for z/OS Program Administration Guide and Reference*.

5

REXX Web connector for MVS -- Commands

Commands are only accepted from Web browsers when the User name entered from a security authority signon window (as in the example in Figure 6 on page 49) matches the value specified in the OWNER parameter specified in the BLMWWEBS command.

Two scenarios exist where security authority signon windows are presented:

- If **AUTHORITY(YES)** is specified, when each browser first accesses the REXX Web connector for MVS through a URL.
- If **AUTHORITY(NO)** is specified, the first time a command is sent to the REXX Web connector for MVS.

BLMWWEBS Commands

BLMWWEBS commands use the following syntax:

```
http://<hostname>:<port>/<cmdprefix string><command>
```

For example

```
http://MVS20:8000/!CLOSE
```

indicates that the host name is MVS20, the Web server is using port number 8000, and ! is the specified command prefix string.

REXX Web connector for MVS Server Commands

These are the commands that can be used with the REXX Web connector for MVS:

CLOSE	Terminate Web server processing.
TRACE	Activate trace. This command has the same effect as specifying TRACE(YES) in the BLMWWEBS command invocation.
NOTRACE	Stop trace. This command has the same effect as specifying TRACE(NO) in the BLMWWEBS command invocation.
EXECFLOW	Start flow trace. This command has the same effect as specifying EXECFLOW(YES) in the BLMWWEBS command invocation.
NOEXECFLOW	Stop flow trace. This command has the same effect as specifying EXECFLOW(NO) in the BLMWWEBS command invocation.

DROP_CACHE	Flush URL cache storage. All static HTML documents present in cache storage are released, counters are reset to zero, and all cache memory in use is freed.
QUERY_CACHE	List all URLs currently in storage as hypertext links.
ECHO	Send the HTTP request just received back to the Web browser. This command is useful when the user wants to check that data is being properly received at the server end, and the server can respond to a trivial request.
DEBUG	Toggles debugging output on and off.

6

REXX Web connector for MVS -- Logging

A log is produced with informational, error, and diagnostic codes. Log output is usually found in the job output data set or SYSOUT.

The following log shows a REXX Web connector for MVS server initially started waiting for the first client browser connection (transaction).

```
97/08/28 10:50:54 050 Process parameter in effect: TCPIP( TCPIP )
97/08/28 10:50:54 050 Process parameter in effect: PORT( 65534 )
97/08/28 10:50:54 050 Process parameter in effect: BACKLOG( 256 )
97/08/28 10:50:54 050 Process parameter in effect: TASKID( IBMUSER )
97/08/28 10:50:54 050 Process parameter in effect: OWNER( IBMUSER )
97/08/28 10:50:54 050 Process parameter in effect: HTML( IBMUSER.INFOWEB.HTML )
97/08/28 10:50:54 050 Process parameter in effect: REALM( INFOWCF )
97/08/28 10:50:54 050 Process parameter in effect: TRACE( NO )
97/08/28 10:50:54 050 Process parameter in effect: EXECFLOW( YES )
97/08/28 10:50:54 050 Process parameter in effect: CMDPREFIX( ! )
97/08/28 10:50:54 050 Process parameter in effect: RECVTIMEOUT( 30 )
97/08/28 10:50:54 050 Process parameter in effect: DOCUMENTROOT( IBMUSER )
97/08/28 10:50:54 050 Process parameter in effect: TIMEZONE( -05:00 )
97/08/28 10:50:54 050 Process parameter in effect: CACHESIZE( 10 )
97/08/28 10:50:54 050 Process parameter in effect: AUTHORITY( NO )
97/08/28 10:50:54 050 Process parameter in effect: LIFESPAN( 30 )
97/08/28 10:50:54 050 Process parameter in effect: PRAGMA( CACHE )
97/08/28 10:50:54 050 Process parameter in effect: DEBUG( NO )
97/08/28 10:50:54 050 Process parameter in effect: SEGMENTSIZE( 16384 )
97/08/28 10:50:54 050 Process parameter in effect: SENDTIMEOUT( 120 )
97/08/28 10:50:54 050 Process parameter in effect: MEDIATYPETABLE(BLM.V1R2M0.SBLMSAMP(BLMWMIME))
97/08/28 10:50:54 050 Process parameter in effect: FIXLEVEL( HOYxxxx )
97/08/28 10:50:54 005 Cache maximum size is 10 megabytes.
97/08/28 10:50:54 024 Routine LOAD_MIME entered, arguments:
97/08/28 10:50:55 024 Routine ZERO_STATS entered, arguments:
97/08/28 10:50:55 024 Routine ZERO_CACHE entered, arguments:
97/08/28 10:50:55 024 Routine LOOP_SOCKET entered, arguments:
97/08/28 10:50:55 024 Routine INIT_SOCKET entered, arguments:
97/08/28 10:50:57 034 Server running InfoWCF/2.0 started
97/08/28 10:50:57 035 Host name is MVSSYS01
97/08/28 10:50:57 036 Host IP address is 9.67.51.15
97/08/28 10:50:57 037 Listening to port 65534
97/08/28 10:50:57 038 Socket number in use is 1
97/08/28 10:50:58 024 Routine WAIT_REQUEST entered, arguments:
```

REXX Web connector for MVS Log Codes

The following is an explanation of the codes which may appear in the REXX Web connector for MVS log.

001 **Cache reset.** *nnn* **bytes released.**

Explanation: The remote client browser successfully issued the !DROP_CACHE command. *nnn* represents the number of bytes that the server has cached and released. This message is displayed only when TRACE(YES) was specified.

Note: In this example, the ! is the command prefix CMDPREFIX(!); you may have a different character for the command prefix.

002 **Socket call** *nnnnnnnn* **produced msg:** *rc*

Explanation: A TCP/IP Socket call *nnnnnnnn* was issued by the server and its return code is displayed. This message is displayed only when TRACE(YES). The return code value can be found in *TCP/IP V3R1 for MVS: Application Programming Interface Reference*

003 **No socket cleanup needed**

Explanation: Informational code to say a socket close is not needed. This message is displayed only when TRACE(YES).

004 **Socket call** *nnnnnnnn* **error msg:** *rc msg*

Explanation: A TCP/IP Socket call was issued by the server and failed. Its return code and any additional information is displayed. The return code value can be found in *TCP/IP V3R1 for MVS: Application Programming Interface Reference*.

005 **Cache maximum size is** *nnnnnnnn* **megabytes**

Explanation: A display of the value for CACHESIZE in effect.

006 **Socket loop**

Explanation: The REXX Web connector for MVS server is waiting for a connection from a client. This information code will periodically be displayed. This message is displayed only when TRACE(YES).

007 **Client at** *mmmmmmmm* **connected using port** *nnnnnnnn*

Explanation: The IP address and port of the current connected client browser are displayed. This message is displayed when TRACE(YES).

008 **New socket number assigned:** *nnnnnnnn*

Explanation: The REXX Web connector for MVS server socket number which has accepted the browser connection is displayed. This message is displayed only when TRACE(YES).

009 **Waiting for more data from** *nnnnnnnn* **port**

Explanation: The REXX Web connector for MVS server socket is waiting for more data from the client browser.

010 **Data read from cache =** *nnnnnnnn* **bytes**

Explanation: If data was cached from a previous transaction such as an HTML, it is sent back directly to the client browser without performing an I/O function. The data length of the item cached is displayed. This message is displayed only when TRACE(YES).

011 **Cache hit ratio is hit_ratio** *mmmm/nnnn* **where** *mmmm* **is the number of cache bytes and where** *nnnn* **is the total number of bytes**

Explanation: The ratio of cache bytes to total bytes is displayed. This message is displayed only when **TRACE(YES)** was specified

012 **Response sent to client at** *nnnnnnn* **where** *nnnnnnnn* **is the client_address.**

Explanation: The REXX Web connector for MVS server sent data to the client browser. This message is displayed only when TRACE(YES).

013 **Server closing**

Explanation: The REXX Web connector for MVS server is closing.

014 **Log and stats**

Explanation: The internal REXX Web connector for MVS server routine to process the log and do internal statistics was called. This can only occur when TRACE(YES) is specified in BLMWWEBS.

015 **Waiting for incoming connection request**

Explanation: The REXX Web connector for MVS server waits for an incoming client browser connection. This message is displayed only when TRACE(YES) was specified.

016 **Reading request data**

Explanation: The REXX Web connector for MVS server has accepted a client browser connection. It now begins the read of the incoming data stream. This message is displayed only when TRACE(YES) was specified.

017 **Processing request**

Explanation: The REXX Web connector for MVS server has read the client browser data. It now starts processing whatever request is in the data. This message is displayed only when TRACE(YES) was specified.

018 **Doing housekeeping**

Explanation: This message indicates internal processing is being done to restore the server to its initial state. This message is displayed only when TRACE(YES) was specified.

019 **Connection to client lost**

Explanation: Transmission of data was interrupted by the client browser. REXX Web connector for MVS server waits for the next client browser connection.

021 **Server restarting. Count number** *mmmm***. Count limit** *5***.**

Explanation: An error condition resulted in the REXX Web connector for MVS server restarting its processing. The current processing iteration and count limit before shutting down are displayed. The count limit is 5.

022 **Error processing URI.** *mmmm nnnnnnnnnnnnnn*

Explanation: The HTTP header response code field is set with code *mmmm* and text *nnnnnnnnnnnnnn* and sent back to the remote client browser.

023 **Directive error. Tag value** *nnnn* **not recognized.**

Explanation: While processing a SERVER directive, the REXX Web connector for MVS server detected an invalid tag value *nnnn* or invalid tag *nnnn*.

024 **Routine** *mmmm* **entered. Arguments** *p1 p2 p3*

Explanation: This informational code displays the routine name and arguments upon entry to an internal REXX Web connector for MVS server routine. This can only occur when EXECFLOW(YES) is specified in BLMWWEBS.

025 **Reading data from** *nnnnnnnn*

Explanation: The REXX Web connector for MVS server attempts to read a data set such as an HTML file.

026 *nnnnnn* **records read.**

Explanation: The REXX Web connector for MVS server successfully read a number of records from a data set such as an HTML file.

027 **File loaded. Elapsed time was:** *nnnnnn*

Explanation: Statistical information giving the time it took to read records from a data set such as an HTML file.

029 **Transaction** *nnnnnnnn* **scheduled.**

Explanation: The number of the current browser transaction. This code comes out before the transaction is processed by the Gateway router.

030 **Gateway completed transaction** *tran_cnt* **elapsed time** *ss.ssssss*

Explanation: The gate completed transaction *tran_cnt* in an elapsed time *ss.ssssss*.

031 **Command** *command* **received from** *user* **address** *address*

Explanation: Command *command* was sent to the REXX Web connector for MVS server by a client with user id *user* and IP address *address*

032 **Error calling gateway router exec.** *mmmmmm nnnnnn*

Explanation: The REXX Web connector for MVS server called the gateway router routine BLMWSWRT to process a form and BLMWSWRT returned with an error condition or was not found. (*mmmmmm* is the error condition and *nnnnnn* is the message returned.) The error condition may be from a forms processing routine which BLMWSWRT called. A message with additional information is also sent to the remote client browser.

033 **Syntax error, code** *nnnn*

Explanation: A REXX syntax error was encountered within BLMWWEBS. Restart the Web connector feature and if the problem persists, save the output and contact the IBM Support Center.

034 **Server running** *nnnnnnnn* **started**

Explanation: The server type is displayed.

035 **Host name is** *mmmmmmmm*

Explanation: The MVS hostname derived from a TCP socket *gethostname* call is displayed.

036 **Host IP address is** *ipaddr*

Explanation: The MVS host IP address from a TCP socket *gethostid* call is displayed.

037 **Listening to port** *nnnn*

Explanation: The port specified for this server is displayed.

038 **Socket number in use is** *nnnn*

Explanation: The socket assigned for the server TCP socket calls is displayed.

050 **Process parameter in effect:** *mmmmmmmm(nnnn)*

Explanation: This informational message displays a parameter and its value passed to the REXX Web connector for MVS server via JCL. If not specified in the JCL, the default value for the parameter is displayed.

051 **Routine name: BLMWWEBS. Environment:***mmmmm*

Explanation: The REXX Web connector for MVS BLMWWEBS routine started under the listed environment *mmmmm* (usually TSO).

055 **Writing data to** *xxxxx*

Explanation: An HTTP put request is being run against the file *xxxxx*

056 *xxxxx* **records filed.**

Explanation: An HTTP put request wrote *xxxxx* records to a file.

057 **File stored. Elapsed time was** *nnnnnn***.**

Explanation: An HTTP put request has completed in the specified elapsed time.

061 **Invalid mode option** *nnnnnn***; specify ASCII or BINARY.**

Explanation: The mime types table input has a syntax error in the translation mode field.

062 **Invalid cache option *nnnnnn*; specify CACHE or NOCACHE.**

Explanation: The mime types table input has a syntax error in the caching mode field.

063 **Invalid media type specified *nnnnnn*.**

Explanation: The mime types table input has a syntax error in the media-type field.

064 **Housekeeping finished. Elapsed time:*nnnnnn***

Explanation: The housekeeping process has completed. The elapsed time is indicated.

065 **Cache cleanup. The following entry released:*nnnnnn***

Explanation: During the housekeeping process, a cache entry was cleared because of the cache utilization exceeding the maximum allowed. This message is issued when the TRACE option is active and helps to track cache utilization.

066 **Unexpected data received when reading request:*nnnnnnnn***

Explanation: This message is issued when the Web browser sends data to the server that is not required to process the HTTP request. The most likely cause of this problem is the use of an additional CRLF sequence that is not required by the HTTP protocol. This protocol violation should be tolerated, though, and the message is mostly information. A CRLF sequence will be reflected as OD25 in EBCDIC. The *nnnnnn* value presents the first 25 bytes of data received in hexadecimal format.

067 **Request header:** *nnnnnn*

Explanation: This message is issued when the TRACE option is active and shows the content of a request header present in the HTTP request being processed.

068 **Request:*nnnnnnnn***

Explanation: This message is issued when the TRACE option is active and shows the HTTP request received from the Web browser and now being processed.

069 **Output from TSO commands:*nnnnnnnn***

Explanation: This message is issued when the TRACE option is active and shows the output from TSO commands issued internally during an HTTP put request process.

7

REXX Web connector for MVS—URL Considerations

Uniform Resource Locators (URLs) represent hypermedia links and links to network services within HTML documents. It is possible to represent nearly any file or service on the Internet with a URL.

The URL can be considered as simply the network equivalent of a file name. The data referred by the URL can exist on any machine on the network, can be served via any of several different methods, and might even be something more complex than a file. URLs can point to queries, to documents stored within databases, to the results of a command, or to whatever the provider decides to send.

Note: URLs accepted by the REXX Web connector for MVS server are not case-sensitive.

Static URLs

With the REXX Web connector for MVS, you can refer to an HTML document stored in a partitioned data set defined by the BLMWSWEB parameter HTML() or, when using the algorithmic URL-to-dataset mapping, to any type of file stored in a partitioned data set member.

Static URLs are maintained by the REXX Web connector for MVS administrator, who has access to the MVS system where the REXX Web connector for MVS is running. Maintenance can be performed directly on the MVS system, for example using the PDF editor to modify an HTML document, or a workstation-based HTML publishing system can be used to create and/or update HTML documents, images, JAVA code, or any other content and store it in the MVS data sets by means of an HTTP PUT request.

The TCP/IP file transfer program (FTP) can also be used to transfer files, such as images, HTML or any other file type, to the MVS system, to be later retrieved using the REXX Web connector for MVS. File transfers should be made using BINARY or ASCII mode depending on the media type of the file being transferred. Please see “The Media Types Table” on page 33.

Static URLs may be cached by the REXX Web connector for MVS server the first time it is accessed, and it will stay in the cache area until the Web server is closed or the DROP_CACHE command is issued (see “REXX Web connector for MVS Server Commands” on page 21).

Caching of a specific URL depends on the attributes assigned to it by the REXX Web connector for MVS. Please see the"The Media Types Table" on page 33 for a discussion of the caching-mode attribute.

Dynamic URLs

Dynamic URLs are generated by the DGA when a request that is mapped to a Forms Processing Routine (FPR) by the router is received.

The FPR accesses the Tivoli Information Management for z/OS database when necessary, performs any calculations required, and formats data to be sent back to the client. This is a real-time operation, and the result is sent to the client as soon as the operation is complete. The same request can generate different HTML output in successive invocations, as the operations are performed against a live database that might have changed between requests.

Such is the case where a URL does not point to a file, but to a database query. In fact, a FPR call can be equivalent to a database query directed to the Tivoli Information Management for z/OS database.

The REXX Web connector for MVS server adds HTTP headers PRAGMA: no-cache and LAST-MODIFIED: to dynamic URLs to prevent caching by proxy servers and Web browsers.

The REXX Web connector for MVS server will expire a document using the value specified on the LIFESPAN unless the form contains the INFOWEB directive, described in "InfoWeb Directive Support - Expiring a Document" on page 33.

Static HTML

Static HTML is HTML source code that is pointed to by a Static URL.

Dynamic HTML

Dynamic HTML is HTML source code that is pointed to by a Dynamic URL.

Dynamic-URL Mapping to a Forms Processing Routine

Any URL that contains the REXX extension or is received as a result of a POST request is considered dynamic and mapped to an FPR. The router REXX program BLMWSWRT is responsible for calling the appropriate FPR associated with the URL. Note that URLs that refer to an FPR need not include the name of the FPR itself; instead, a codename is mapped to the FPR by the router. The URL should contain this name as a unique identifier. One FPR can be mapped to many different codenames by the router, but only one FPR can be associated with any given codename. The codename is obtained from the URL by eliminating the leading protocol, host and path information and the trailing extension. For example, if the URL was `http://mvs20:8000/DEMO/CODE/CREATE.REXX` the prefix `http://mvs20:8000/DEMO/CODE/` is eliminated, as is the REXX extension, leaving `CREATE` as the codename to be referenced.

This mapping process is used by the DGA router routine to translate URLs to FPR references. If the router does not explicitly define a codename to be accepted as part of a dynamic URL, the REXX Web connector for MVS server will reject the request and inform the user accordingly.

This mode of operation is required to prevent users from running code from anywhere in the MVS system. They can only run REXX code from the SYSEXEC or SYSPROC concatenations in the REXX Web connector for MVS JCL.

Static-URL to Data Set Mapping

The REXX Web connector for MVS provides support for an algorithmic URL-to-data set mapping. The algorithm is backwards-compatible with the method used in previous releases.

In order to convert a URL (Universal Resource Locator) to an MVS data set name, the following elements are taken into account by the mapping algorithm:

- The DOCUMENTROOT parameter of the BLMWWEBS command (see 13 for additional information about this parameter.)
- The URL specified in the Web browser HTTP request.
- The media types table, described in "The Media Types Table" on page 33. This table is used to assign certain attributes to the data based on the extension given. Extension is the last component of the URL; for example, in the URL `http://mvs20:8888/demo/test.html` the extension is `HTML`, which indicates that the data is *HTML* source. Following the same mechanism, an GIF extension would indicate an image in *gif* format, an MPEG extension would indicate a movie in *mpeg* format, and so on. The media types tables provides the information required to properly identify the content-type based on the extension.
- The use of the INFOWEB marker in the URL. URLs with this marker will be mapped using the same mechanism used in earlier releases of this product. In this case, the URL will have the format `<http>://<host:port>/INFOWEB/<name>.<HTML>`. If the extension is omitted, then `HTML` is substituted. The URL will be mapped to a member in the partitioned data set specified in the HTML parameter of the BLMWWEBS command. The member name is obtained from the URL by eliminating the leading protocol, host and path information and the trailing extension. For example, if the URL was `http://mvs20:8000/INFOWEB/DEMO/CODE/CREATE.HTML.` the prefix `http://mvs20:8000/DEMO/CODE/` is eliminated, as is the HTML extension, leaving `CREATE` as the member name to be referenced. For example, in the case of `http://mvs20:8001/INFOWEB/blmwhdbm.` there must be a member named BLMWHDBM in the HTML data set for this URL to be accepted by the REXX Web connector for MVS
- If the URL does not contain the INFOWEB marker as in the previous case, a mapping algorithm will be used. The algorithm will combine the value of the DOCUMENTROOT parameter of the BLMWWEBS command (see 13 for a description of this parameter) and data derived from the URL specified. Consider the general format of the URL as `http://<host:port>/qual-1/qual-2/qual-3/.../qual-n/name.ext.` the REXX Web connector for MVS will take the value of the DOCUMENTROOT parameter, and use it as the initial portion of the target data set name, concatenate all the qualifiers using a period as a separator, use the extension as a suffix and finally use the name specified as the member to be referenced. The data set name obtained will be translated to uppercase to conform with MVS naming conventions. `http://<host:port>/qual-1/qual-2/qual-3/.../qual-n/name.ext.` will translate to `documentroot.qual-1.qual-2.qual-3.....qual-n.ext(name)`

  Note: The document root value provides additional security to the REXX Web connector for MVS by restricting the data set names that can be referenced by an

HTTP request. For example, if the URL is `http://mvs20:8888/demo/version1/mainpnl.htmland` the document root parameter value is `INFOMAN.WEB.CONNECTOR` then the data set name generated by the algorithm is `INFOMAN.WEB.CONNECTOR.DEMO.VERSION1.HTML(MAINPNL)`

The fact that the URL is converted to an MVS data set name effectively propagates those limitations on what names and qualifiers can be used for the REXX Web connector for MVS URLs.

Note: The REXX Web connector for MVS will not create a new partitioned data set as a result of a PUT request, but it will create a new member if required.

When creating the partitioned data sets that will be referenced by the Web connector, you should follow some allocation standards. “Allocation Partitioned Data Sets To Be Used with the REXX Web connector for MVS” contains additional information regarding possible standards.

Allocation Partitioned Data Sets To Be Used with the REXX Web connector for MVS

The REXX Web connector for MVS will read or write data to MVS partitioned data sets. The recommended data control block (DCB) parameters to be used for the allocation are as follows: For binary (octet-stream) data, you should use a minimum logical record length of 8192 bytes (LRECL=8192), and a blocked, variable record length format (RECFM=VB). For ASCII (text) data you should use a logical record length larger than the longest record that you intend to store. A recommended value is 1024. The record format should also be blocked, variable record length (RECFM=VB). If you intend to edit this data using MVS editors (such as the ISPF/PDF editor) the maximum record length accepted by the editor may be a limiting factor. If you intend to edit your HTML code using workstation editors only, you can use any logical record length value up to the maximum accepted by your MVS system.

Include Directive Support

The REXX Web connector for MVS provides two forms of Server Side Include (SSI) directive support. The syntax is the following:

```
<!--#INCLUDE FILE=name.extension -->
```

where *name* is the name of a static HTML source member from your HTML partitioned data set or

```
<!--#INCLUDE VIRTUAL=/qual-1/qual-2/.../qual-n/name.extension -->
```

where */qual-1/qual-2/.../qual-n/name.extension* is the fully qualified name (also called the *virtual path*) of a static HTML document.

Static and Dynamic HTML source is scanned for *include* directives by the REXX Web connector for MVS server before sending response data to the Web browser client.

Nested *include* directives are supported by the REXX Web connector for MVS server code.

InfoWeb Directive Support - Expiring a Document

The REXX Web connector for MVS provides a server side directive to expire an HTML document. Static and dynamic HTML source is scanned for the INFOWEB directive by the REXX Web connector for MVS server before sending the response data to a Web browser client. In the following examples, the directive, tag, and its value can be uppercase or lowercase. The actual operation depends on the browser and the options chosen by the user that relate to the method by which the browser manages it cache.

<!--#INFOWEB EXPIRES=NOW-->

When the directive **NOW** is used, the server will place **EXPIRES** and **PRAGMA:NOCACHE** into the HTTP header. If supported by the browser, this will cause the browser to request the form be re-sent from the server instead of being retrieved from the cache.

<!--#INFOWEB EXPIRES=NEVER-->

When the directive **NEVER** is used, the server will not place **EXPIRES** and **PRAGMA:NOCACHE** into the HTTP header. This will prevent the browser from requesting the form to be re-sent. The browser will use the copy from its cache, if available.

<!--#INFOWEB EXPIRES=nnnnnn-->

nnnnnn is a numeric value in the range 1–9999999 and is the number of minutes the form is considered valid. After the specified number of minutes has elapsed, the browser may request the form be re-sent. Until then, the browser can use the form from its cache.

The Media Types Table

The primary use of the media types is to assign a content-type to a Uniform Resource Locator (URL) based on the extension specified in the URL. Two other attributes are assigned using the media types table: one is the translation mode, and the other is the caching mode.

- `EXTENSION` is the key to the table. It is obtained from the URL being processed. Examples of extensions are `HTML`, `GIF`, `JPEG`, `MPEG`, and `CLASS`.
- `CONTENT-TYPE` specifies what data typing applies to the URL. For example, for a GIF image, the content-type is `IMAGE/GIF`. For HTML source the content-type is `TEXT/HTML`. The syntax of the content type field is:

  ```
  <type>/<subtype>
  ```

  Parameters as defined by the HTTP 1.0 protocol are not supported. *Content-type* is also called *media type* and *mime type*.

Many media types are defined by Internet standards maintained by the Internet Assigned Number Authority (IANA). See the RFC 2048 for more information on how the registered media types are administered. Note that the HTTP protocol recommends following the standards when applicable, but does not restrict the use of unregistered content-types. As long as the client and server can successfully interpret the content-type exchanged, the use will be valid.

Translation Mode

specifies if the data received should be converted from ASCII to EBCDIC or vice versa. The two possible values for this field are ASCII and BINARY.

ASCII implies translation, and is commonly used for text. For example, for a content-type of text/html, the translation mode could be ASCII, in which case MVS-based editors can be used on the data. For a content-type of image/gif, the translation mode should be BINARY, because the data should not be modified using MVS-based tools.

Caching mode

specifies whether the data could be cached by the Web connector internal caching mechanism.

Note: The caching mode is not used to influence caching by the Web browser itself. This attribute influences behavior at the server side, not at the client side. To modify the behavior of the client caching algorithm, see "InfoWeb Directive Support - Expiring a Document" on page 33.

The format of the table is as follows:

- Any line that starts with an asterisk (*) or a slash asterisk (/*) is treated as a comment.
- All other lines relate an extension to a media type. There are four fields per line. Each field is made up of one single token. The position of the field in the line is not important, although it is convenient to maintain fields aligned in columns. The first field is the extension, the second the content-type, the third is the translation mode, and the fourth is the caching mode. Any data following is considered a comment.

This is a copy of the sample table provided in SBLMSAMP(BLMWMIME):

```
*
* Media Type table.
*
* The format of this table is as follows:
* Any line that starts with an asterisk (*) is a comment.
* All other lines identify a mime type. There are four fields per line
* with input data. Each field is made up of one single token.
* The positioning of the field in the line is not important,
* although it is convenient to maintain fields aligned in columns.
* Here is the meaning of each data field:
*
* 1 - Extension name
*            The extension name is used as the key to determine
*            the media type.
*            Extension names can have any value.
*
* 2 - Media Type (also called content-type and mime-type)
*            The media type identifies what kind of data is present
*            Media types are defined by internet standards maintained
*            by the IANA, Internet Assigned Number Authority. See
*            the RFC 2048 for more information on how the media types
*            are administered. The list presented here is not
*            exhaustive, if any new media type is required, a new
*            entry should be added to this table.
*
* 3 - Translation mode
*            Indicates if the data should be converted
*            from EBCDIC to ASCII or viceversa.
*            Should be coded as binary (no translation) or ASCII.
*
* 4 - Caching mode
*            Indicates if the data can be cached by the
*            web connector.
*            The primary use of this field is to allow for
```

```
*            caching of static HTML. Binarys are usually cached
*            by the browser. Generally speaking, the only extensions
*            that should be cached are HTML, HTM and TXT.
*
* Any data after the fourth field is considered a comment.
* All the fields are case-insensitive.
*
* Change Activity:
*  $L1=DCR521    HOYT100 970721 DMGPGTR: OS/390 web connector
*
*Ext  Media-Type                    Mode   Cache?  Comments
*---- ------------------------------ ------ ------- --------------------

aif   audio/aiff                    binary nocache // AIFF
aifc  audio/aiff                    binary nocache // AIFF
aiff  audio/aiff                    binary nocache // AIFF
art   image/x-jg                    binary nocache //
au    audio/basic                   binary nocache //
avi   video/avi                     binary nocache //
bin   application/octet-stream      binary nocache // Unspecified binary
class application/x-java            binary nocache // java class
crt   application/x-x509-ca-cert    binary nocache // certificate
css   text/css                      binary nocache //
der   application/x-x509-ca-cert    binary nocache // certificate
dll   application/x-msdownload      binary nocache //
eml   message/rfc822                binary nocache // Internet Mail Message
ps    application/postscript        binary nocache //
eps   application/postscript        binary nocache //
ai    application/postscript        binary nocache //
exe   application/x-msdownload      binary nocache //
fif   application/fractals          binary nocache //
gif   image/gif                     binary nocache //
gz    application/x-gzip            binary nocache //
hqx   application/mac-binhex40      binary nocache //
htm   text/html                     ascii    cache //
html  text/html                     ascii    cache //
iii   application/x-iphone          binary nocache //
ins   application/x-internet-signup binary nocache // x-internet-signup
isp   application/x-internet-signup binary nocache // x-internet-signup
jpg   image/jpeg                    binary nocache // JPEG
jfif  image/jpeg                    binary nocache // JPEG
jpeg  image/jpeg                    binary nocache // JPEG
jpe   image/jpeg                    binary nocache // JPEG
js    application/x-javascript      ascii    cache // Javascript source/code
latex application/x-latex           binary nocache // LATEX
man   application/x-troff-man       binary nocache //
mov   video/quicktime               binary nocache // Quicktime Video
movie video/x-sgi-movie             binary nocache //
mpg   video/mpeg                    binary nocache // MPEG
mpe   video/mpeg                    binary nocache // MPEG
mpeg  video/mpeg                    binary nocache // MPEG
mp2   video/mpeg                    binary nocache // MPEG
enc   video/mpeg                    binary nocache // MPEG
mpa   video/mpeg                    binary nocache // MPEG
m1v   video/mpeg                    binary nocache // MPEG
nsc   application/x-conference      binary nocache //
nws   message/rfc822                binary nocache // Internet News Message
qt    video/quicktime               binary nocache //
ram   audio/x-pn-realaudio          binary nocache // RealAudio
ra    audio/x-pn-realaudio          binary nocache // RealAudio
rpm   audio/x-pn-realaudio-plugin   binary nocache //
sit   application/x-stuffit         binary nocache //
snd   audio/basic                   binary nocache //
tar   application/x-tar             binary nocache //
tgz   application/x-compressed      binary nocache //
tif   image/tiff                    binary nocache //
tiff  image/tiff                    binary nocache //
```

```
txt   text/plain                    ascii    cache // Text - 7 bit
uls   text/iuls                     binary nocache //
wav   audio/wav                     binary nocache //
xbm   image/x-xbitmap               binary nocache //
z     application/x-compress        binary nocache //
zip   application/x-zip-compressed  binary nocache //
323   text/h323                     binary nocache //
```

8

REXX Web connector for MVS and REXX Web connector for OS/390 -- REXX Globals

The REXX Global Variable (RGV) Service is shipped as part of the REXX Web connector for MVS and as part of the REXX Web connector for OS/390 so that Database Gateway Applications can share REXX variables among several REXX routines.

The sample Database Gateway Application (DGA) uses the RGV service to define a global space name.

Once this is defined, the DGA issues RGV calls to store and retrieve REXX variables from this global space.

RGV Service Invocation

REXX EXECs may create, update, read, write and drop RGVs. Provision is made for reading and writing multiple variables in a single function call. Variables may be specified generically by suffixing an asterisk * to their name.

Names for RGVs conform to the same naming rules as for REXX variables. When an RGV is accessed, a REXX variable of the same name will be set to the value of the RGV. Similarly, when setting a global variable, the value will be taken from a REXX variable of the same name.

There is no limitation on values, value lengths, names, or name lengths other than those limitations imposed by REXX.

RGVs are grouped into “variable spaces”, each of which will be referred to by a name specified by the user.

The space name may be any combination of characters up to 15 in number. More than one variable space may be in use at one time. There is no interaction between variable spaces, so they may contain similarly named variables. This concept permits instances of sets of variables of the same name to be maintained.

Function Call Syntax

The RGV interface is implemented as a group of external REXX function calls.

All calls return a REXX result of the form:

```
<rc> <message>
```

where <*rc*> is a numeric code indicating the level of success and <*message*> is a qualifying message.

Codes signify level of success and are not grouped according to reason.

Functions

BLMXSMK

Create an RGV variable space.

Syntax: result=blmxsmk(<space_name>,<level>)

Input:

<Space Name>	User-defined name 1 to 15 characters.
<Level>	TASK | JOBSTEP

Return:

RC 0 => success
RC 4 => space already exists
RC 12 => parameter error(s)
RC 20 => system service failure

BLMXSDR

Destroy a RGV space and all variables assigned to it.

Syntax: result=blmxsdr(<space_name>,<level>)

Input:

<Space Name>	User-defined name 1 to 15 characters.
<Level>	TASK | JOBSTEP

Return:

RC 0 => success
RC 4 => space does not exist
RC 12 => parameter error(s)
RC 20 => system service failure

REXX variables are not updated by this function call.

BLMXVPT

Write one or more REXX variables to an RGV space.

Syntax: result=blmxvpt(<space_name>,<level>,<var_name1>,<var_name2>,.)

Input:

<Space Name>	User-defined name 1 to 15 characters.
<Level>	TASK | JOBSTEP
<var_name1>..	List of REXX variable names. Names may be specified generically by suffixing with an asterisk (*).

Return:

RC 0 => success
RC 4 => REXX variable is uninitialized
RC 8 => space does not exist

RC 12 => parameter error(s)
RC 20 => system service failure

BLMXVGT

Read one or more global variables and set them to a REXX variable.

Syntax: result=blmxvgt(<space_name>,<level>,<var_name1>,<var_name2>,..)

Input:

<Space Name>	User-defined name 1 to 15 characters.
<Level>	TASK | JOBSTEP
<var_name1>..	List of REXX variable names. Names may be specified generically by suffixing with an asterisk (*).

Return:

RC 0 => success
RC 4 => A global variable does not exist
RC 8 => space does not exist
RC 12 => parameter error(s)
RC 20 => system service failure

BLMXVDR

Drop one or more RGV variables and free the storage they occupy.

Syntax: result=blmxvdr(<space_name>,<level>,<var_name1>,<var_name2>,..)

Input:

<Space Name>	User-defined name 1 to 15 characters.
<Level>	TASK | JOBSTEP
<var_name1>..	List of REXX variable names. Names may be specified generically by suffixing with an asterisk (*).

Return:

RC 0 => success.
RC 4 => A global variable does not exist
RC 8 => space does not exist
RC 12 => parameter error(s)
RC 20 => system service failure

REXX variables are not updated by this function call.

Using RGV Services -- an Example

To illustrate how a REXX application would use the RGV service, we will look at two of the sample DGA forms that are provided, BLMWFCRT (Create record) and BLMWSFIN (Initialize HLAPI/REXX).

The SPACENAME parameter global is passed as a parameter to BLMWFCRT; in addition, BLMWFCRT issues the RGV service BLMXVGT which retrieves all variables from the pool of Global Variables that begin with blg* (the * is the “wildcard” indicator):

```
result = blmxvgt(GLOBAL,'JOBSTEP','blg*')
parse var result result result_text
```

The result is checked for a failure condition. If the failure condition is that the RGV service has not been initialized (result = 8), then **BLMWSFIN**, the initialization form REXX routine, is invoked.

```
if result > 8 then return '400' 'Global variable error.' result_text;
if result = 8 then do;
call BLMWSFIN(GLOBAL);    /* API not initialized */
```

BLMWSFIN initializes the HLAPI/REXX API and calls BLMXSMK so that an RGV space can be started.

```
/** Initialize RGV **/
result = BLMXSMK(GLOBAL,'JOBSTEP')
```

BLMWSFIN then stores the blg* variables set by the HLAPI/REXX in the REXX global space by calling BLMXVPT. The one variable needed by other REXX form routines is BLG_ENVP. Control returns to the calling routine BLMWFCRT.

```
result = blmxvpt(GLOBAL,'JOBSTEP','blg*')
```

BLMWFCRT now retrieves the blg* global variables by issuing the BLMXVGT service. This occurs whether the BLMWSFIN routine was entered on the present or a previous Web browser transaction.

```
result = blmxvgt(GLOBAL,'JOBSTEP','blg*')
```

For the purposes of this sample, BLMWFCRT needs only the REXX variable BLG_ENVP restored prior to calling the HLAPI/REXX.

9

The Database Gateway Application

Overview of the Database Gateway Application

The Database Gateway Application (DGA) provides the gateway services to your system database.

The Database Gateway Application consists of REXX programs, known as forms processing routines (FPRs), and HTML documents.

The operation, interface, and means of modifying the DGA are described in this chapter. Also described are the REXX Web connector for MVS and the REXX Web connector for OS/390 service routines (which are shipped as source), as well as their relationship to the DGA components.

The Database Gateway Application operates according to the protocol of whichever Web server called it. The entire Database Gateway Application template can be used as is, or can be modified or enhanced to fit your needs. The source for the service routines supplied by the REXX Web connector for MVS and the REXX Web connector for OS/390 can also be modified for your application needs.

Figure 5 on page 42 illustrates how the REXX Web connector for MVS and the REXX Web connector for OS/390 each interface with the DGA and service routines.

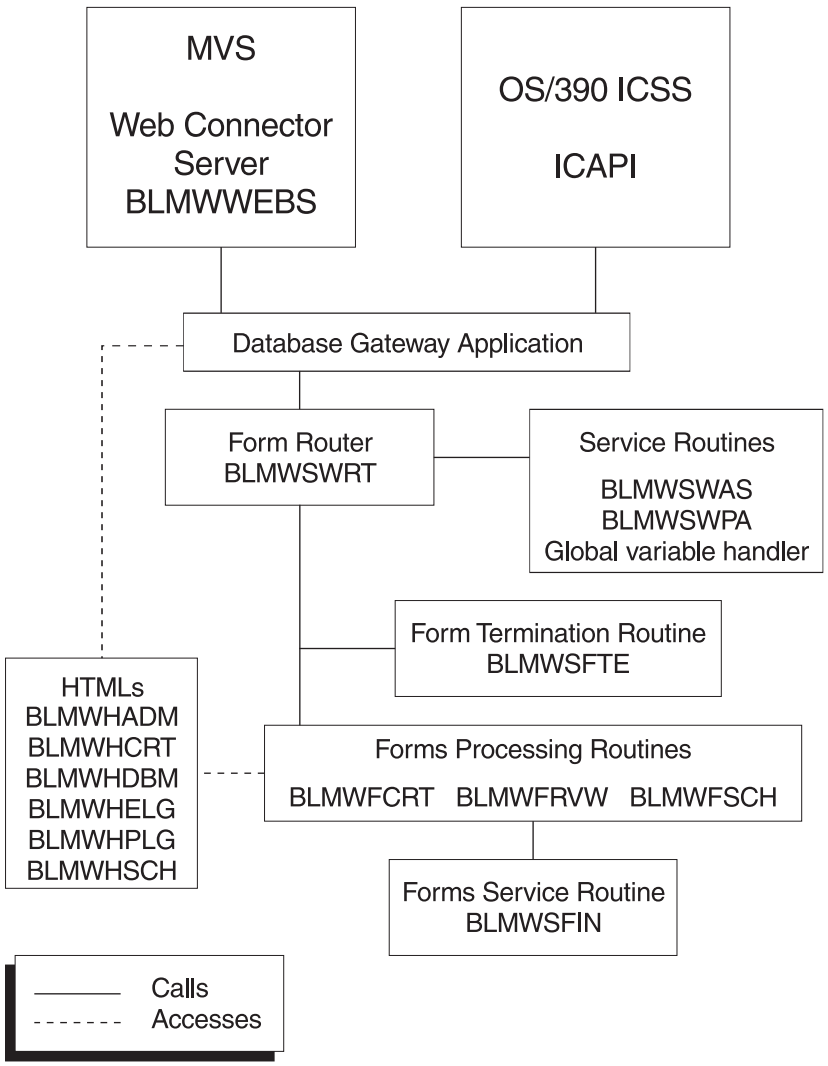


Figure 5. Web connector Interfaces

DGA REXX Forms Processing Routines (FPRs)

The forms processing routines (FPRs) use REXX code, REXX HLAPI calls, and custom services provided by the service routines to process HTTP requests received from client machines using Web browsers as their Graphical User Interface (GUI). Forms processing routines have different characteristics, depending on which environment, the REXX Web connector for MVS or the REXX Web connector for OS/390, called them:

REXX Web connector for MVS
The forms processing routines are passed key environment variables. Some of the overhead of processing environment data is simplified by this Web server; for example, form input data is contained in a single environment variable, and there is no need to process the data according to its method of GET or PUT. The REXX QUEUE statement is used to return HTML data.

REXX Web connector for OS/390
The forms processing routines are HTTP Server for OS/390 GWAPI REXX programs which utilize GWAPI services to access environment variables, set HTTP content fields, and return HTML data to the client browser.

How DGA REXX Forms Processing Routines (FPRs) Are Invoked

The sample DGA forms processing routines are called by the Web server router module BLMWSWRT when an HTML form is processed from a remote browser and received as a HTTP data stream by the Web server. Normally, a call to a DGA forms processing routine is initiated by a form tag within the HTML processed by the remote browser. For example, the Search HTML sample BLMWHSCH has the form tag:

```
<form action=SEARCH.REXX method=GET>
```

When your Web browser processes the BLMWHSCH HTML, the form action **SEARCH.REXX** and any input fields are sent as an HTTP request string to the Web server (either the REXX Web connector for MVS or the REXX Web connector for OS/390). The Web server recognizes that a REXX forms processing routine is to be invoked, and passes control to the form router BLMWSWRT to call the **Search forms** processing routine.

Forms processing routines may also be invoked from HTML hyperlinks, as in the case of the **List records opened today** option from the BLMWHDBM HTML:

```
<a href=SEARCH.REXX?TODAY=YES>
List records opened today</a>
```

In this example, the **SEARCH.REXX** is processed in the same way by the REXX Web connector for MVS, meaning the Search forms processing routine is called. The **?TODAY=YES** is user input, which is passed to the forms processing in the BODY parameter for the REXX Web connector for MVS; for the REXX Web connector for OS/390 server, the BODY is extracted from an HTTP environment variable QUERY STRING.

The REXX Web connector for MVS Server Service Router - BLMWSWRT

The routine BLMWSWRT is invoked by the REXX Web connector for MVS server and the REXX Web connector for OS/390 when an FPR is to be called. While the routine name and called interface must always remain intact, the rest of the processing can be modified by the user.

REXX Web connector for MVS interface

The following parameters are passed to BLMWSWRT:

URL The HTTP URL (Universal Resource Locator) passed unmodified from the remote client browser.

MEMBER The REXX routine name in the URL. The sample router translates this member name to an actual MVS/ESA REXX FPR and calls it.

BODY The body portion of the URL being processed. This is the string following the ? character. For example, if the URL is:

```
SEARCH.REXX?TODAY=YES
```

the body parameter would be:

```
TODAY=YES
```

To facilitate parsing in FPRs, the & character which denotes separation between HTTP name/value pairs is translated to a X'FF'.

CLIENT The REXX Web connector for MVS client address, port number, and client domain in the form:

```
CLIENTAD:clientport clientdomain
```

USER If **AUTHORITY(YES)**, the user field specified from the remote client browser authentication window is decoded by the security exit BLMWSWSE and passed to the DGA. If **AUTHORITY(NO)**, the user field is null.

TRANCNT The number passed to the DGA as a unique identifier for the transaction.

REQ_HEADER
The HTTP request header sent by the remote client browser.

ATABL The ASCII character set table returned from a call to the BLMWSWAS routine.

SERVER The REXX Web connector for MVS server address and port number in the form yyyyyy.zzzz where yyyyyy is the server address and zzzz is the port number.

REXX Web connector for OS/390
No parameters are passed for the REXX Web connector for OS/390. A subset of the variables listed above for the REXX Web connector for MVS are extracted using the IMWXRD call described in described in the *HTTP Server Planning, Installing, and Using V5.2 for OS/390* (look for information about writing GWAPI programs).

BLMWSWRT Operation

The DGA template provided maps the URL passed to an actual FPR and calls that routine.

REXX Web connector for MVS
The FPR should inform the REXX Web connector for MVS server of any error condition by returning data using the REXX statement, as follows:

```
RETURN code message
```

Where
code Normal completion or error from the forms processing routine or router.
message A message string or null.

REXX Web connector for OS/390
REXX routines invoked by BLMWSWRT may return non-zero return codes, but BLMWSWRT must always pass back a return code of 200 to the REXX Web connector for OS/390 server.

Sample DGA REXX Forms Processing Routines

The REXX Web connector for MVS and the REXX Web connector for OS/390 provide the following sample forms processing routines to illustrate how a DGA application can be constructed:

BLMWFCRT Create record routine

BLMWFRVW
View record routine

BLMWFSCH Search record routine

DGA REXX Forms Processing Routines Interface

Each of the above routines shares a common set of parameters upon invocation by the REXX Web connector for MVS router BLMWSWRT (these were more fully described on page 43):

BODY The body portion of the URL being processed (see page 43).

USER The user field specified (see page 44).

SERVER The REXX Web connector for MVS server address and port number (see page 44).

ICAPI A flag indicating whether the DGA is running in a REXX Web connector for MVS environment or a REXX Web connector for OS/390 environment

SPACENAME The name of the space for RGV invocations.

DGA REXX Forms Processing Routines Operation

Each of the forms processing routines processes a specific Tivoli Information Management for z/OS transaction requested by the remote client browser. Each performs the following:

- Uses the REXX global function to retrieve and store shared REXX variables such as retrieving the REXX HLAPI environment variable BLG_ENVP (see "REXX Web connector for MVS and REXX Web connector for OS/390 -- REXX Globals" on page 37 for additional information on REXX globals).
- Parses the input data (BODY) passed by the remote client browser and sets input REXX HLAPI data variables. For example the parse statement in BLMWFCRT

```
Parse upper var body 1 'S0B2D='s0b2d(ff) ,
1 'S0B9B='s0b9b(ff) ,
1 'S0BE7='s0be7(ff) ,
1 'S0B59='s0b59(ff) ,
1 'S0E0F='s0e0f(ff)
```

  searches and sets matching s-word variables with values from the input parameter body.
- Calls the REXX HLAPI to submit the requested transaction. The subroutine INFO_API accomplishes this.
- Returns data to the remote client browser.

REXX Web connector for MVS

For normal completion of a transaction, data and messages are queued on the REXX stack, and a return code of **201** is set. The return statement is always of the form

```
RETURN code message
```

Where:

code Normal completion or error.

message A message string or null. Normal completion output is queued on the REXX stack.

Error conditions return with an error code such as 502 with an error message string; for example:

```
RETURN   502   The DESCRIPTION field is blank
```

REXX Web connector for OS/390
Data is returned to the remote client browser by the IMWXWRT call described in the *HTTP Server Planning, Installation, and Using V5.2 for OS/390* (look for information about writing GWAPI programs).

Sample DGA REXX Forms Service Routines

DGA REXX Forms Service Routine BLMWSFIN Interface

The REXX Web connector for MVS and the REXX Web connector for OS/390 provide a sample forms service routine which can be called by any of the other forms processing routines. This sample forms service routine is BLMWSFIN, and its function is to initialize the REXX HLAPI session and create REXX globals space. The following information is passed to BLMWSFIN:

Spacename The name of the space for RGV invocations.

The return statement for BLMWSFIN is of the form

```
RETURN code message
```

If the code is non-zero, a message is returned to the caller. You can change the HLAPI/REXX parameters to conform to the standards of your installation. For example, in BLMWSFIN, you should change the following values:

```
application_id='IBMUSER'
session_member='BLGSES00'
privilege_class='MASTER'
```

Other control variables such as **pidt_name**, **separator_character**, **text_option**, and **text_medium** are contained in the FPRs BLMWFCRT, BLMWFRVW, and BLMWFSCH. You may want to modify these values in the FPRs to meet your needs.

DGA REXX Forms Service Routine BLMWSFTE Interface

Another routine which can be called is BLMWSFTE. This routine is called for the REXX Web connector for MVS server when the server is terminated by the !CLOSE command or is explicitly invoked by a browser for the REXX Web connector for OS/390 server. The purpose of this routine is to perform Tivoli Information Management for z/OS cleanup and any required application cleanup before the server terminates processing. Failure to use this routine to properly terminate the Tivoli Information Management for z/OS environment may cause the Web server to abend.

The following information is passed to BLMWSFTE:

ICAPI A flag indicating whether the DGA is running in a REXX Web connector for MVS environment or a REXX Web connector for OS/390 environment

Spacename The name of the space for RGV invocations.

Maximum Number of Global Variable POOLS
This equates to the maximum number of concurrent sessions.

DGA return codes

The following codes are used by HTTP FPRs to indicate the success or failure of a request and provide some information about the cause of the error to the browser client. The code is placed in the HTTP response code field.

200 Normal completion.

201 HTTP Created
400 Error, used for REXX global variable service failures
500 Error, internal service error, such as the forms service routine BLMWSFIN failed
502 Error, general DGA failure such as errors from the REXX HLAPI

HTML Documents

Sample HTML documents provided with Tivoli Information Management for z/OS are:

BLMWHCRT - (Create)	Create a new problem management record
BLMWHDBM - (Menu)	A menu of choices which serves as a sample REXX Web connector for MVS home page
BLMWHELG - (Epilog)	A standard epilogue for all HTML documents
BLMWHPLG - (Prolog)	A standard prologue for all HTML documents
BLMWHSCH - (Search)	Display a list of search criteria
BLMWHSAM - (Validation)	Sample HTML for validation samples
BLMWHADM - (Link)	Link to the ADSM Web Client

Web Server Service Routines

These routines may be called by either the REXX Web connector for MVS or by the REXX Web connector for OS/390, and provide functions such as ASCII translation to the Web server. The naming convention used is BLMWSWxx.

Those provided with this release are:

BLMWSWAS	Web Service return, returns ASCII to EBCDIC conversion table
BLMWSWGM	Web Service GMT conversion
BLMWSWPA	Web Service, provides translation for URL-encoded data to the normal EBCDIC character set (the URL-encoded algorithm is defined by the HTTP 1.0 standard). It is used to convert data received from the browser client to readable format.

Modifying the Database Gateway Application

Once you have reviewed the DGA program source, you are ready to change it. Here are some suggestions to help you.

For the REXX Web connector for MVS

- Copy the DGA routines and HTML members to another partitioned data set (PDS) for your testing. In this example, this is called TESTWEB.PDS.
- Choose a TCP/IP port number. Update the PORT operand in your JCL to reflect the new number.

  Note: Each active REXX Web connector for MVS Web server must use a different TCP/IP port. Client machines use the port number to select a specific Web server with which to communicate.

- Copy the sample JCL BLMWJCL, update it as needed, and submit as a batch job (this process is explained in "Running the REXX Web connector for MVS Server as an MVS Batch Job" on page 7).
- Develop the HTML forms you want to use by using an HTML editor at your workstation or by working directly with the HTML code in your HTML PDS and copying the forms to your MVS data set.
- If you are using HTML forms processing, you must code an HTML form processing routine. Test the routine code independently, then copy the tested routine into TESTWEB.PDS. If you were to use the name MYTEST as the subroutine name, the hypertext link to invoke the forms processing routine would look like this:

```
<a href=MYTEST.REXX?Test</a> Test the new code
```

For the REXX Web connector for OS/390

- Create a new directory and copy the HFS DGA routines and HTML to this directory for your testing. In the following example, this directory is called TESTWEB.
- Update IBM HTTP Server for OS/390 configuration directives so that TESTWEB will be picked up.
- Develop the HTML forms in your HTML Hierarchical File System (HFS) directory.
- If you are using HTML forms processing, you must code an HTML form processing routine. Test the routine code independently, then copy the tested routine into TESTWEB. If you were to use the name MYTEST as the subroutine name, the hypertext link to invoke the forms processing routine would look like this:

```
<a href=MYTEST.REXX?Test</a> Test the new code
```

Sample Database Gateway Application

The sample DGA depicted on the following pages uses Web Explorer as a browser. The screens you see may be different if you are using some other browser.

You can start the sample application by issuing the following command from a browser window:

```
http://hostname:portnumber/INFOWEB/BLMWHDBM.html
```

where *hostname* is the host name of your MVS system and *portnumber* is the port number used to access the IBM HTTP Server for OS/390.

If you are using REXX Web connector for MVS, and have specified AUTHORITY(YES) or you enter a BLMWWEBS command, the first panel presented is the "security" signon screen. If you have an HTTP Server for OS/390 access control directive defined (Document Protection and Protection setups), a security signon screen similar to this will appear.

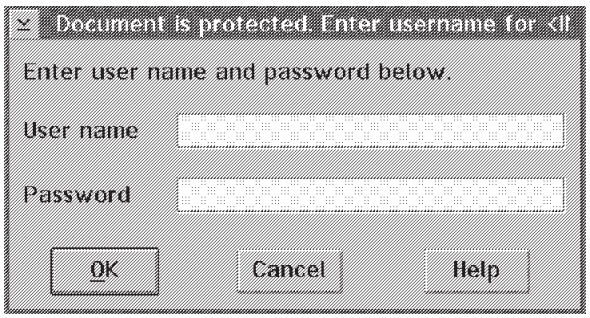


Figure 6. Security Signon Screen

Note: When you are using a BLMWWEBS command, the USER NAME field is compared to the OWNER parameter specified in BLMWWEBS ("BLMWWEBS Parameters" on page 10). It is case-sensitive.

The sample application provides several choices:

- If you select **List records opened today**, a list of records opened today is displayed.
- If you select **Search for records**, Figure 8 on page 50 is displayed
- If you select **Create a new problem record**, Figure 9 on page 51 is displayed.
- If you select **ADSM Web Client** (Tivoli Storage Manager), Figure 10 on page 52 is displayed.
- If you select **Validation Samples Using Java Applets and JavaScripts**, the validation process using supplied applets is started. This process is described in "Overview of the Sample Programs" on page 54.

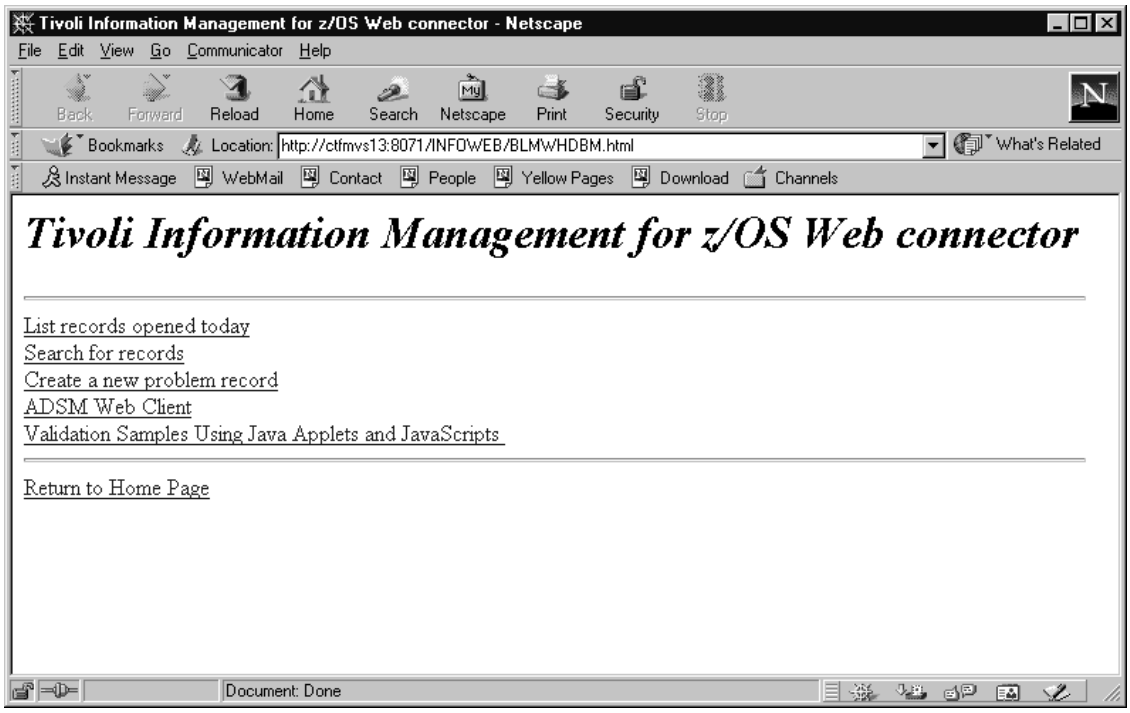


Figure 7. Web Connector Feature Home Page

If you selected **Search for records** on the Web connector feature home page, Figure 8 displays selectable search criteria to search for records.

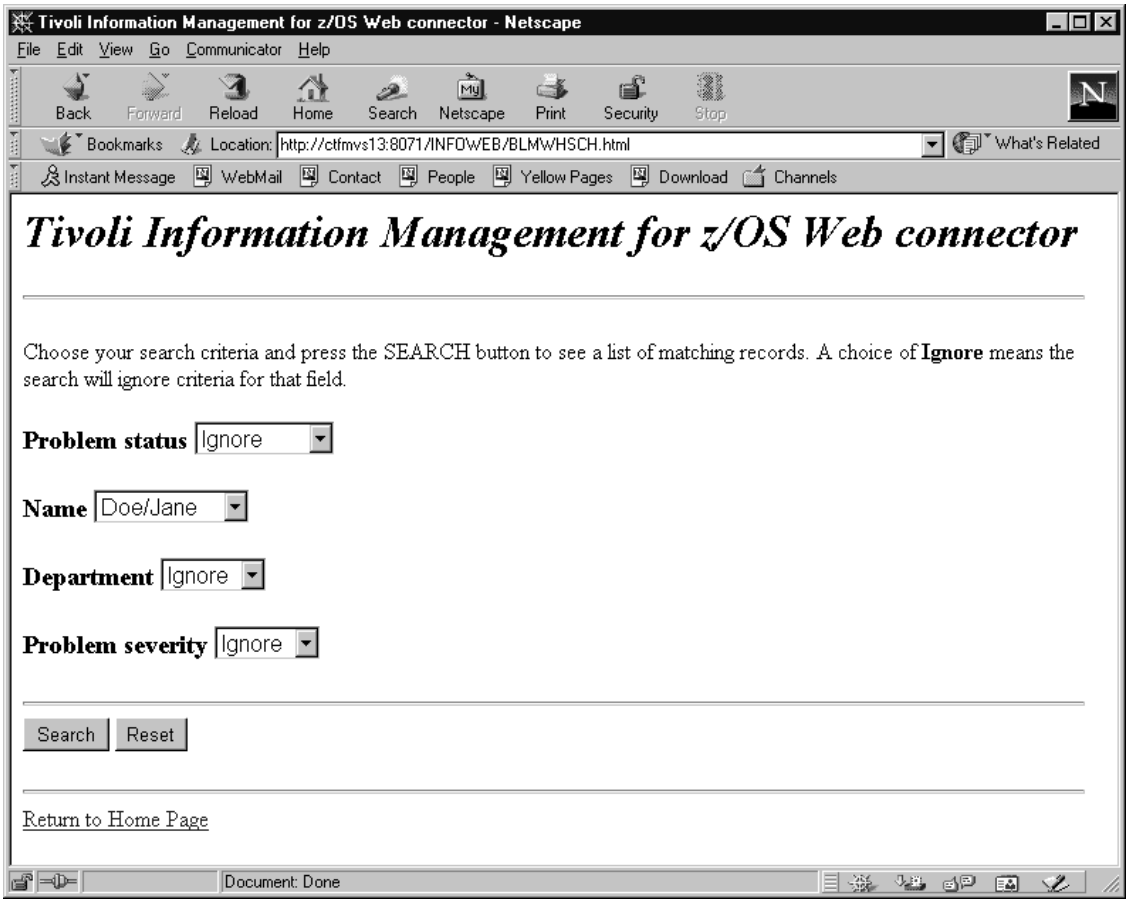


Figure 8. Search for Records

If you selected **Create a new problem record** on the Web connector feature home page, Figure 9 on page 51 is displayed. From this panel, the user can enter information to create a new record.

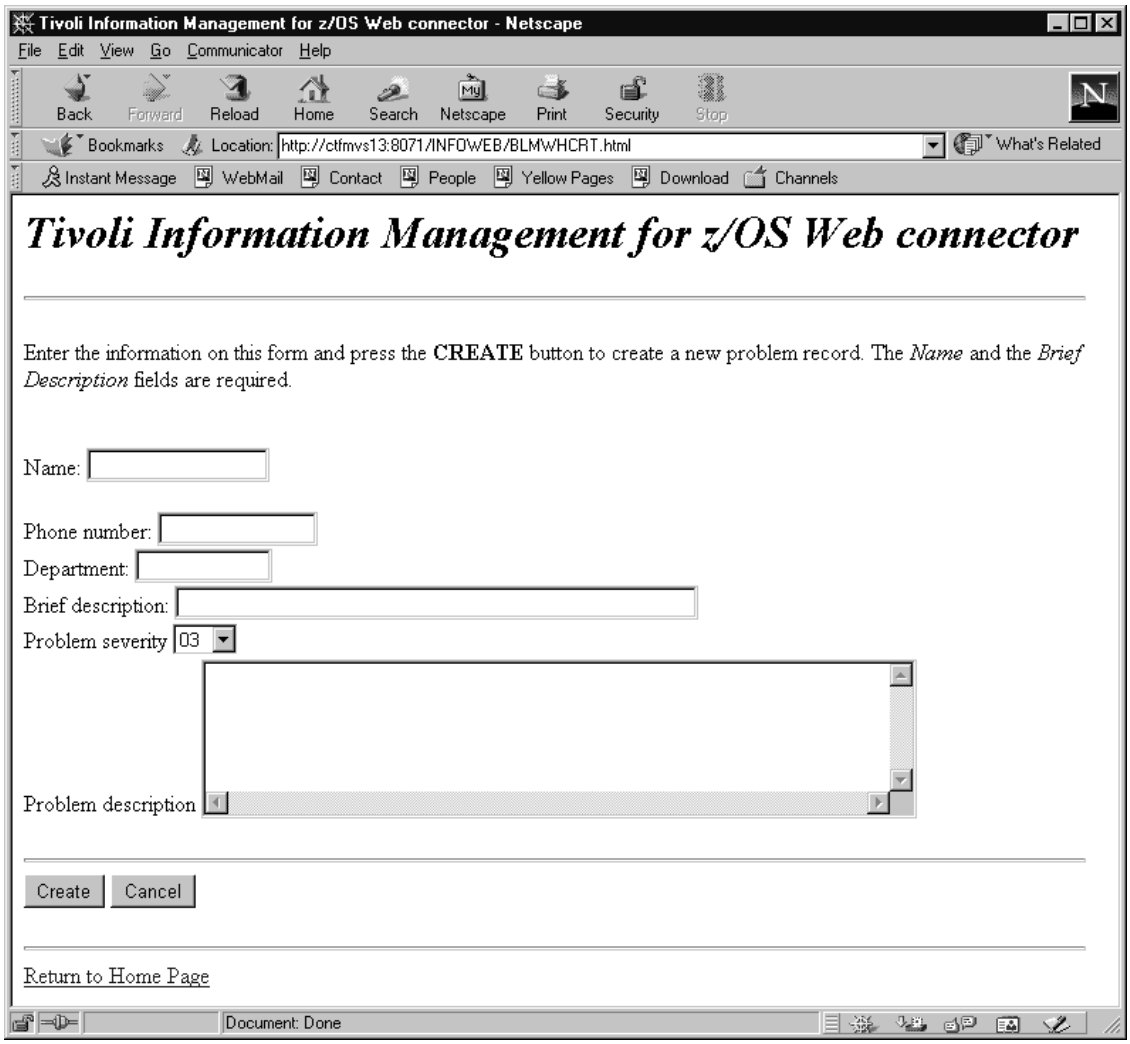


Figure 9. Create a Record

If the record is created successfully, a confirmation message is displayed.

If you selected **ADSM Web Client** on the Web connector feature home page, Figure 10 on page 52 is displayed. From this panel, you can provide information that will link to the ADSM Web Client (Tivoli Storage Manager) interface.

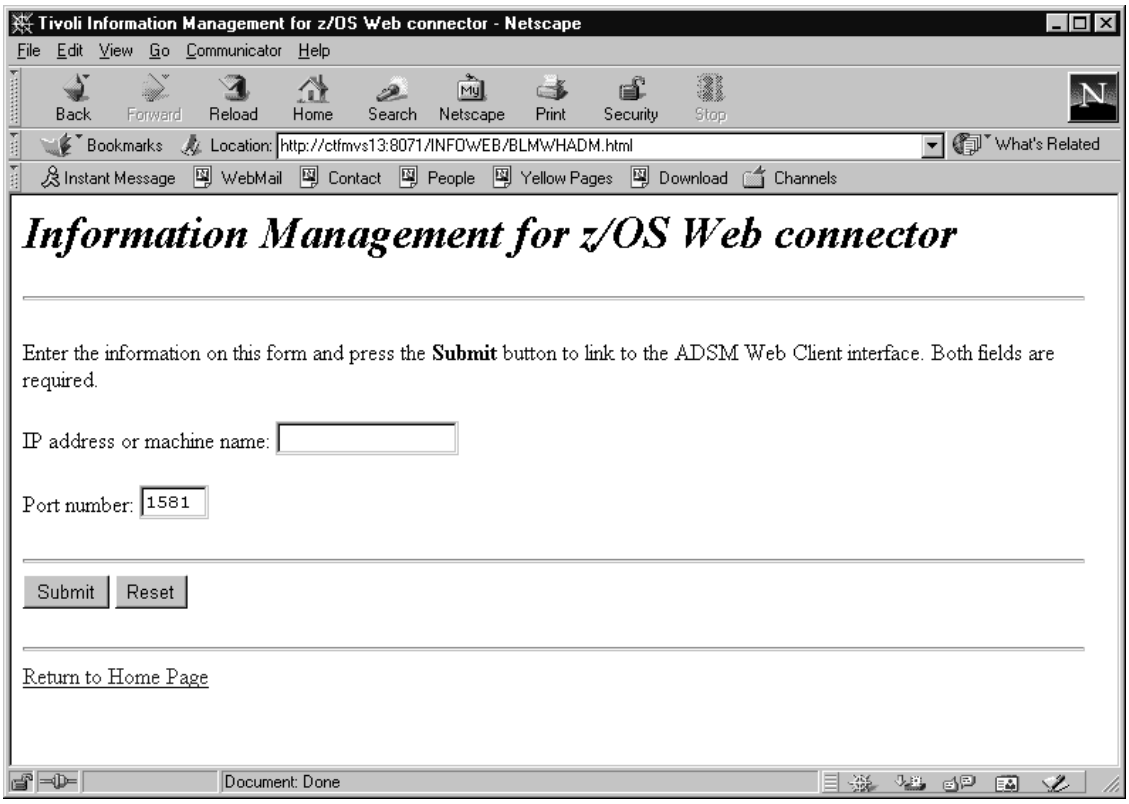


Figure 10. Link to ADSM Web Client Interface

10

Using Java and JavaScript to Validate Data Fields

Data Validation on the Server

When you use the REXX Web connector for MVS or the REXX Web connector for OS/390, validation of user-entered data is performed within the Tivoli Information Management for z/OS database which resides on the MVS or OS/390 server. If you have a form containing fields which need to be validated, the server would validate each field; if any of the fields contained invalid data, the server returns an error to the user, who would then be required to resubmit the data. These are the steps involved when data validation for a CREATE record is done on the server:

1. Data entered by a user is sent to the Web connector server.
2. The Web connector server relays the data entered to a Tivoli Information Management for z/OS Application Program Interface (API); the API transmits the data to the Tivoli Information Management for z/OS database, where the data is validated using a robust validation pattern algorithm to determine if the data is valid.
3. The API returns a response to the server.
   - If the data is valid, the response indicates that a record was successfully created.
   - If the data is not valid, the API returns return code BLG_RC and reason code BLG_REAS.
4. The server passes the message to the client's browser via HTTP.

This method of data validation on the server is good as long as the user's input data is valid. But if the user enters data that is not valid (such as a name field containing a character that is not among the recognized characters), the user must wait for the server to validate the data, detect the error, and return the error message to the user. The user must then correct the data and resubmit it.

Data Validation on the Client Using Java Applets

Tivoli Information Management for z/OS provides a methodology to perform data validation on the "Client" rather than on the "Server". This methodology, which uses a combination of Java and JavaScript, may significantly improve the usability and performance of data validation. Using such a technique, the user is provided feedback almost instantaneously about the data entered before anything is sent to the Web server for processing. Three sample programs which utilize Java "applets" are provided. The validation methods provided by these programs use the actual validation patterns from Tivoli Information Management for z/OS for each field passed by the Client browser. The method also knows which fields are required fields.

Overview of the Java Applets

Two Java applets are provided to allow data field validation based on validation patterns. One applet is the Storage applet Blmwjast, which keeps a table of field keys and their respective validation patterns. Another applet is the Validation applet Blmwjafv, which does the actual validation of the user's input data based on the supplied validation patterns.

Validation that Is Supported

These are the types of validation which can be performed by using the supplied Java applets:

- The supplied Java applets support these data validation characters: A, B, C, I, N, Rnn, S, Vnn, and X (for a complete explanation of these validation characters, see the *Tivoli Information Management for z/OS Panel Modification Facility Guide*).

  Note: The validation character L is accepted, but left-padding does not occur.
- The ability to identify which field is a required field.
- The ability to identify whether too much or too little data was entered as it pertains to the pattern specified.

Validation that Is Not Supported

These types of validation are not performed by the supplied Java applets:

- Case-sensitive validation.
- DBCS validation.
- Left zero padding of input data is not provided. The applet does check to see if the required number of characters is entered.
- In search operations, neither the asterisk * nor the period . are supported.
- Equal type patterns (for example =DATE) are ignored.
- Multiple response fields.
- The Tivoli Information Management for z/OS string type field maximum data length is not validated.

Overview of the Sample Programs

All of the sample programs are samples of a Problem Record Create HTML form. Each of the sample programs uses the supplied Java applets to validate input data. If the user selected **Validation Sampler Using Java Applets and JavaScripts** from the Web Connector Home Page, Figure 7 on page 49, the Sample Programs Menu is displayed:

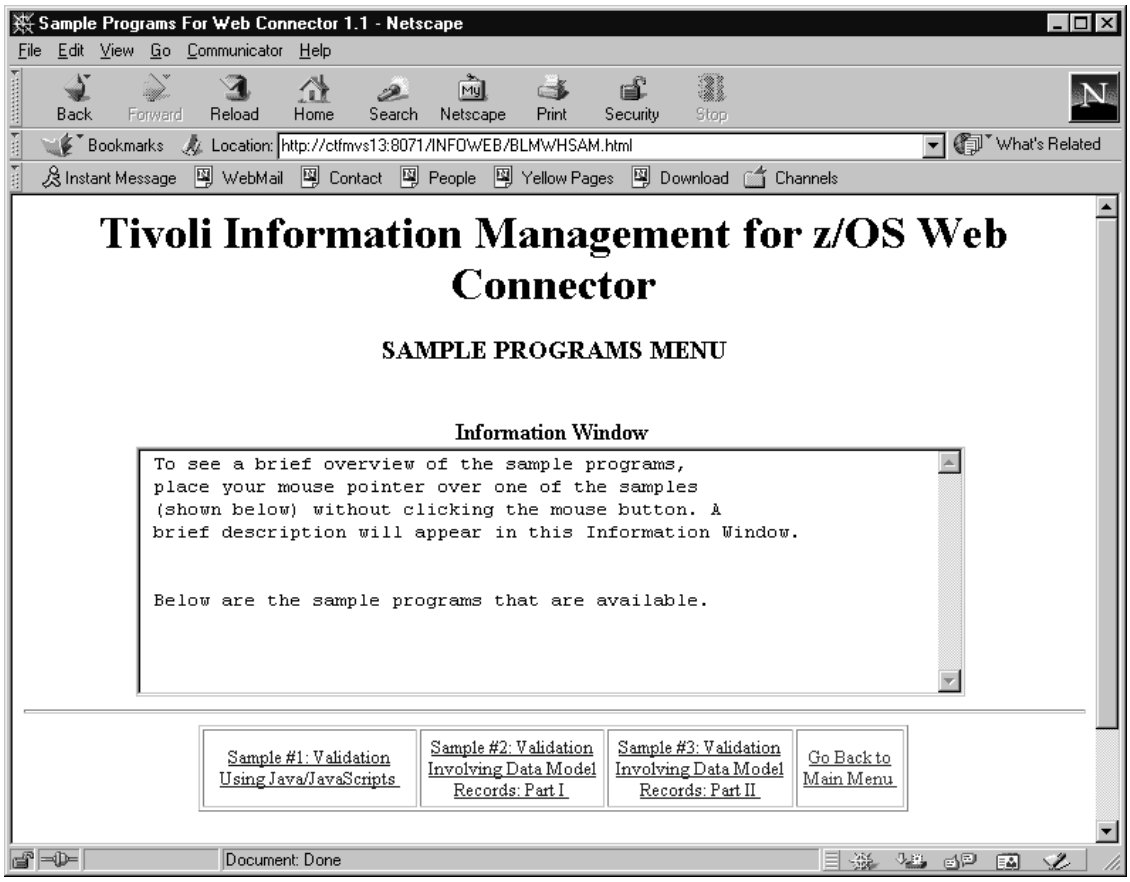


Figure 11. Sample Programs Menu

Sample # 1—Data Field Validation Using Java and JavaScripts

This sample program illustrates the use of the Java applets and client JavaScripts to validate data. The sample consists of an HTML form and some JavaScript code that provides the communication between the user and the Java applets. The information used to validate the input data is predefined in the HTML form. If the user selected **Sample #1: Validation using Java/JavaScripts** from the Sample Programs Menu, Figure 11, the following panel is displayed:

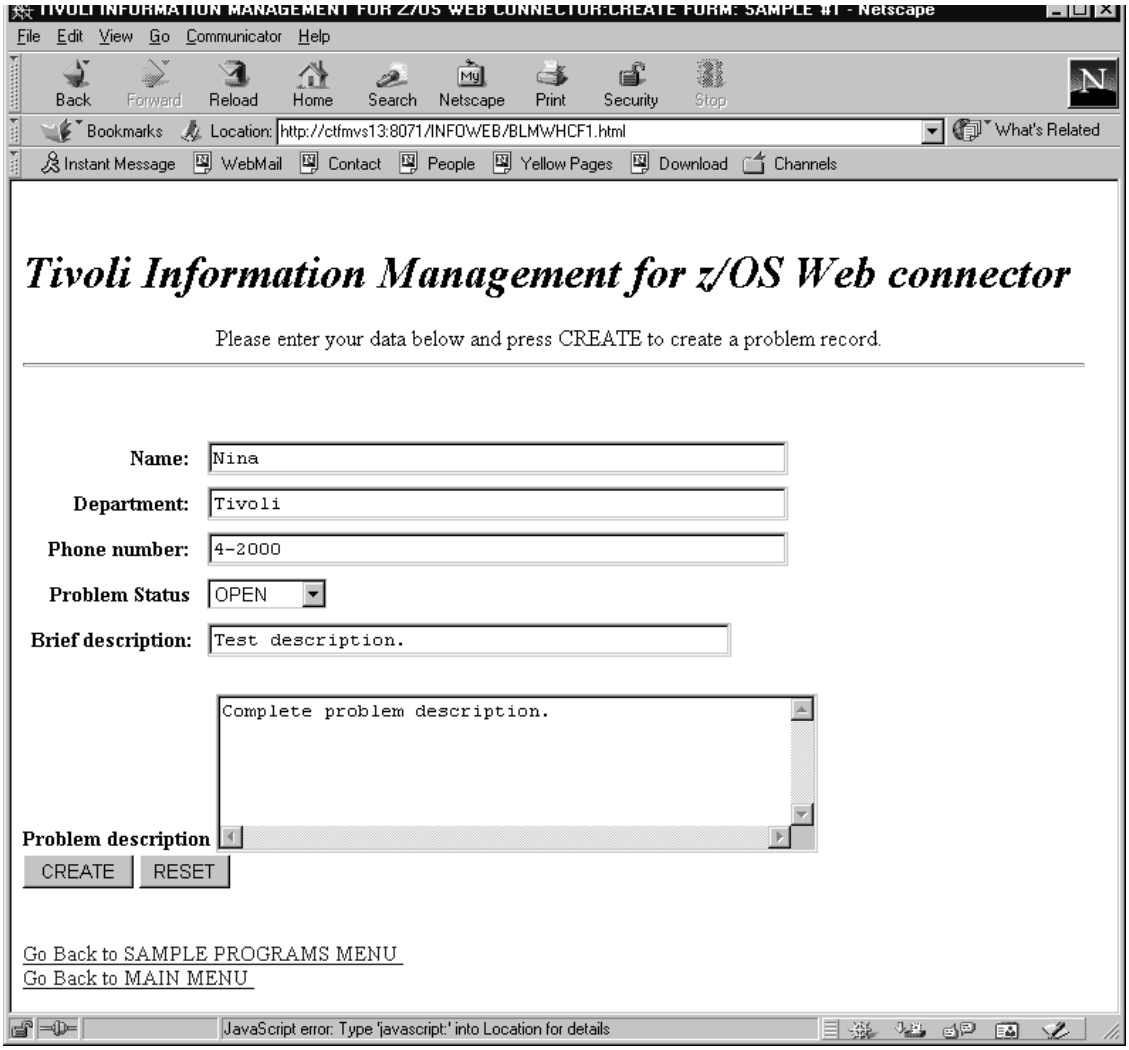


Figure 12. Using Data Field Validation to Create a Problem Record

Sample # 2—Data Validation: Dynamically Generated HTML Forms

In Sample # 2, the validation information is not specified in the HTML form. Instead, the HTML form is dynamically generated by the Sample #2. Information needed to build the form and to validate input data is obtained from data view records and data attribute records.

Sample # 3—Data Validation: Static HTML Forms

In Sample # 3, while the validation information is still built dynamically from the data model records, the HTML form is not. The HTML form is static, which allows the user to use any HTML form so long as the information used for validation is located in the data model records.

Java Applet Prerequisites

The prerequisites for using the Java field validation methodology are:

- REXX Web connector for MVS or REXX Web connector for OS/390.
- A client Web browser that supports both Java Version 1.1.8 or higher and JavaScript calling applets compiled with Version 1.1.8 or higher.

- A Web server that is capable of serving Java applets.
- The supplied Java applets stored on a Web server (see Step 2 below).

Installation and Configuration of the Sample Programs

The following procedure is used to install the necessary files:

1. Install either the REXX Web connector for MVS or the REXX Web connector for OS/390.
2. Copy these files from the \web\java_aps directory on the provided CD-ROM to a Web server where Java applets can be accessed:
   - Blmwjast.class
   - Blmwjafv.class
3. Ensure that these files reside in the data set where the HTML files reside on your REXX Web connector for MVS server or your REXX Web connector for OS/390 server:
   - BLMWHCF1
   - BLMWHCIN
   - BLMWHJS1
   - BLMWHNJS
   - BLMWHSAM
   - BLMWHUCF
   - BLMWHCPR
4. Modify BLMWHJS1 to specify the correct URL of your Web server where the Java applets reside. In the following example, note that CODEBASE= lines occur on two separate lines. Change http://yourWebServer/java in each line so that it contains the URL of your Web server.

```
<APPLET NAME="STApplet" CODE="Blmwjast.class"
CODEBASE="http://yourWebServer/java"
WIDTH=5 HEIGHT=5 >
</APPLET>
<APPLET NAME="ValidateApplet" CODE="Blmwjafv.class"
CODEBASE="http://yourWebServer/java"
WIDTH=5 HEIGHT=5 >
 </APPLET>
```

5. Ensure that these files reside in the data set where your REXX files reside on your REXX Web connector for MVS server or your REXX Web connector for OS/390:
   - BLMWCRTV
   - BLMWFCVD
   - BLMWFCV1
   - BLMWREVD
6. Modify both BLMWFCVD and BLMWFCV1 to specify the correct URL of your Web server where the Java applets reside. In the following example, note that the CODEBASE= lines occur in two locations. Change each line so that it contains the URL of the Web server.

```
queueit '<applet name=STApplet code=Blmwjast.class'
queueit 'codebase=http://yourWebServer/java/'
queueit 'width=5 height=5 >'
queueit '</applet>'
queueit '<applet name=Validate code=Blmwjafv.class'
queueit 'codebase=http://yourWebServer/java/'
queueit 'width=5 height=5 >'
queueit '</applet>'
```

7. A TSX BLMWSAMD is provided in the SBLMTSX data set. It will create a data view record and some data attribute records for use by Sample #2 and Sample #3. (You may want to refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* and the *Tivoli Information Management for z/OS Application Program Interface Guide* for additional information about data view records, data model records, and data attribute records.) The TSX BLMWSAMD will use the assisted entry panels for the fields to "prime" the attribute records. All attribute records must contain an s-word index. BLMWSAMD supplies an s-word index for those fields whose assisted entry panels specify "Collect s-word from caller=YES". The following records are created:

   DVSAMPLE Data view record containing all of the sample attribute records

   AAA#REQN Data attribute record for reporter name

   AAA#RQDP Data attribute record for reporter department

   AAA#PHON Data attribute record for reporter phone

   AAA#STAT Data attribute record for problem status

   AAA#PRII Data attribute record for priority

   AAA#DSAB Data attribute record for brief description

   AAA#DTXT Data attribute record for description text

   In order to run BLMWSAMD, do the following:

   a. Verify that your Tivoli Information Management for z/OS session allocates the dd name SBLMTSX. SBLMTSX must allocate the partitioned data set which contains BLMWSAMD.

   b. BLMWSAMD assumes an immediate response chain (irc) of ;INITIALIZE,3,2 flows to the Management application and from there a selection of **5** will start record create and flow to BLG00000. If this is not true, copy the TSX to a local TSX data set (which must be allocated to DD name BLGTSX when running Tivoli Information Management for z/OS) and modify the immediate response chain and selection number (look for the phrase `CHANGE THE FOLLOWING:` in BLMWSAMD).

   c. From the Tivoli Information Management for z/OS command line, type **;RUN BLMWSAMD** to run the TSX.

   d. Verify successful completion by displaying record DVSAMPLE, selecting option **3**, and verifying that the correct attribute record IDs are contained in the list.

8. Now you are ready to try out the sample programs. Start your REXX Web connector for MVS server or your REXX Web connector for OS/390 server.

9. On your Web browser, specify the correct URL location of your REXX Web connector for MVS server or your REXX Web connector for OS/390 server, for example, **http://mvssystem:80/infoweb/blmwhsam.html**. From the resulting screen, select sample 1, which will provide you the opportunity to test the data validation. If you provide

invalid data (for example, a tilde ~ character in the reporter name field), an error message will advise you that you entered an invalid character; but the error message is issued from the client workstation where the data validation occurred rather than from the server.

The Supplied Java Applets

Two Java applets which are used in the process of performing validation on the client rather than on the server are provided on the CD-ROM. The Storage applet, Blmwjast.class, and the Validation applet, Blmwjafv.class, work together to perform validation. Data validation relies on field keys and Tivoli Information Management for z/OS validation patterns. Figure 13 illustrates how the Storage applet works with the Validation applet to provide data validation on the client.

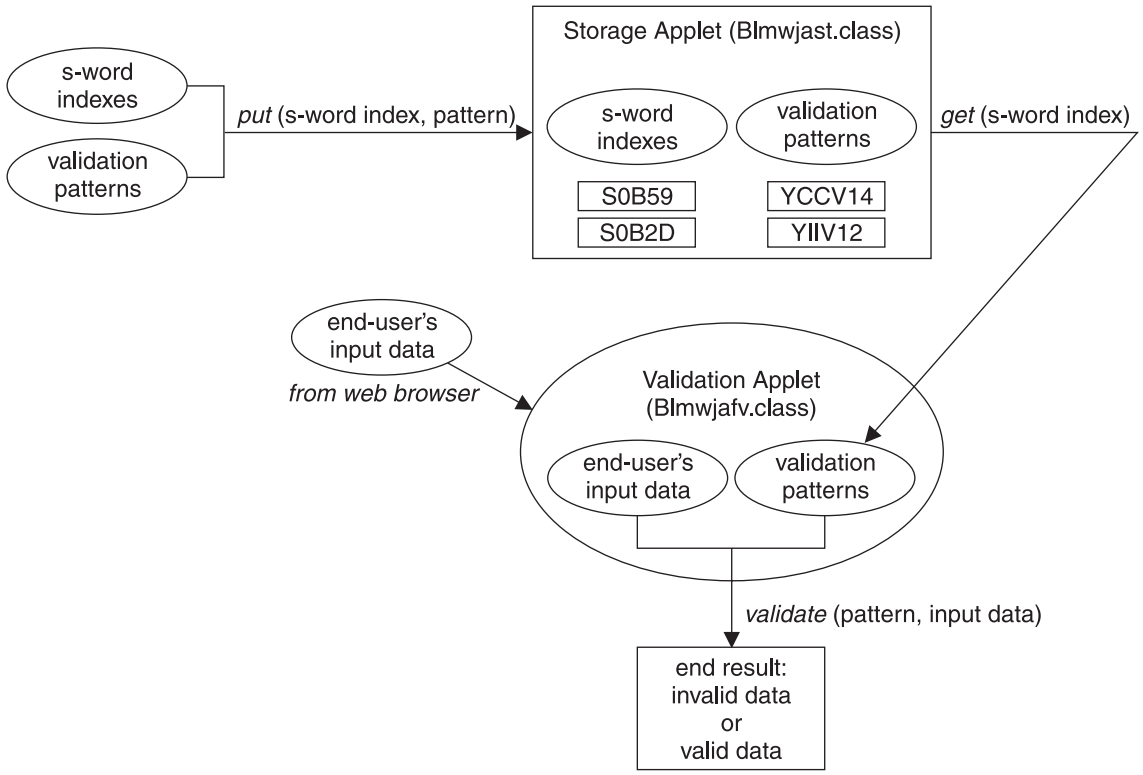


Figure 13. Data validation using the supplied Java applets

The Storage applet takes as input the s-word index and validation pattern and maintains a table of all the s-word indexes and their corresponding validation patterns on the client machine. Whenever there is data to be validated, the Validation applet gets the validation pattern from the Storage applet, and validates the input data using that pattern. The Validation applet sends the result back to the HTML form, where JavaScripts bind the HTML form and the applet to present the result to the user. If the data is valid, there is no further interaction between the form and the user. If the data is invalid, JavaScripts provide the communication media between the HTML form and the user. The user will see a JavaScript Alert Window with the applet-generated error message in it.

This is a description of the functions provided by each of the supplied applets:

The Storage applet: Blmwjast.class

public void put(String key, String value)

This function puts the s-word indexes as the keys and the corresponding validation patterns as the values into a storage table. If there is no pattern involved with the s-word index, then specify **"_NO PATTERN"** as the pattern string. The pattern must be preceded by the required reply field indicator:

Y for a required field

_ for a non-required field or not sure that it is required

An example of this function call is

- `put("S0B59","YCCV14")`

which will put the s-word S0B59 with a corresponding pattern YCCV14 in the storage table. Some other examples of this function call are

- `put("S8100","_ACR5,NCR5")`
- `put("S0BEE","Y{INITIAL},{OPEN},{CLOSED}")`
- `put("S0E0F","_NO PATTERN")`

public String get(String key)

This function returns the pattern string from the storage table based on an s-word index. An example of this function call would be

```
X=get("S0B59")
```

which will assign the pattern YCCV14 to the variable X (assuming that the put function of the previous step `put("S0B59","YCCV14")` has already been done).

The Validation applet: Blmwjafv.class

public String validate(String pattern, String userData)

This function validates the user's input data based on a provided validation pattern. It will return the result as a string. If the user's input data is valid, `true` is returned; otherwise, an error message is returned. An example of this function call would be

```
y=validate(X,"Doe/Jane")
```

This assumes from the previous example call that the variable `X` contains the pattern string `YCCV14`; the variable `y` would contain the return string as `true`.

Some other items of which you should be aware are:

- Literal patterns must be enclosed in braces {}
- A 1000-character limit exists for patterns

In addition to these Java applets, JavaScripts have been provided which bind the HTML form and the Java applets, making the form interactive. JavaScripts provide a two-way communication between Java and HTML. To invoke a Java function that is contained in a Java class, the following JavaScript format is used:

```
document.yyyyy.zzzzz
```

where yyyyy is the name of the applet which was assigned in the HTML <applet> tag and zzzzz is the function call. For example, if you named your Storage applet STApplet, this is how to make a put of an s-word index and a validation pattern:

```
document.STApplet.put('S0B59','YCCV14')
```

or

```
document.STApplet.put("S0B59","YCCV14")
```

The Supplied Samples

Sample #1—Data Field Validation Using Java and JavaScripts

This sample mainly involves two HTML files, BLMWHJS1 and BLMWHCF1. In this sample, the s-word indexes and patterns provided to the Storage applet are coded in the HTML form, BLMWHCF1. The s-word index is coded as the NAME parameter of the HTML form field and the pattern is coded as the default VALUE parameter of the HTML form field. Because the patterns are coded as the default VALUEs, the patterns will temporarily appear in the HTML form at the beginning of the HTML page download. (The feature is designed this way because it is assumed that most users are more familiar with HTML than with JavaScript.) Below is a section of code from the HTML form file, BLMWHCF1, that illustrates where the s-word index and pattern are defined:

```
<TR>
   <TD ALIGN="right"><B>  Name: </B>
   </TD>
   <TD ALIGN="left">   <input type="text"
     NAME="S0B59"
     VALUE="YCCV14"
     maxlength=50>
     size=50
   </TD>
</TR>
```

The HTML file, BLMWHJS1, contains the JavaScript code that take the values from the HTML form file and gives them to the Storage applet. The JavaScripts communicate to the Validation applet when the CREATE button is pressed by passing the correct patterns and user's input data to the Applet.

Sample #2—Data Field Validation: Dynamically Generated HTML Forms

This program dynamically generates the HTML form for you based on data model records. Data model records provide an alternative to assisted entry panels to store data composition of your records; additional information on data model records can be found in the *Tivoli Information Management for z/OS Panel Modification Facility Guide* and the *Tivoli Information Management for z/OS Application Program Interface Guide*. Attribute records must contain an s-word index. This program also provides the s-word indexes and patterns for the Java applets. Two REXX programs, BLMWREVD and BLMWFCVD, and one Terminal Simulator Exec are involved in this sample. The Terminal Simulator Exec generates a data view record and six data attribute records in your Tivoli Information Management for z/OS database which are used by this sample. BLMWREVD is the REXX program that performs the API call to retrieve data view and data attribute records which contain the s-word indexes and validation patterns. BLMWFCVD is the REXX program that

loads the Java applets and builds the HTML form and JavaScripts based on what BLMWREVD obtained from your Tivoli Information Management for z/OS database.

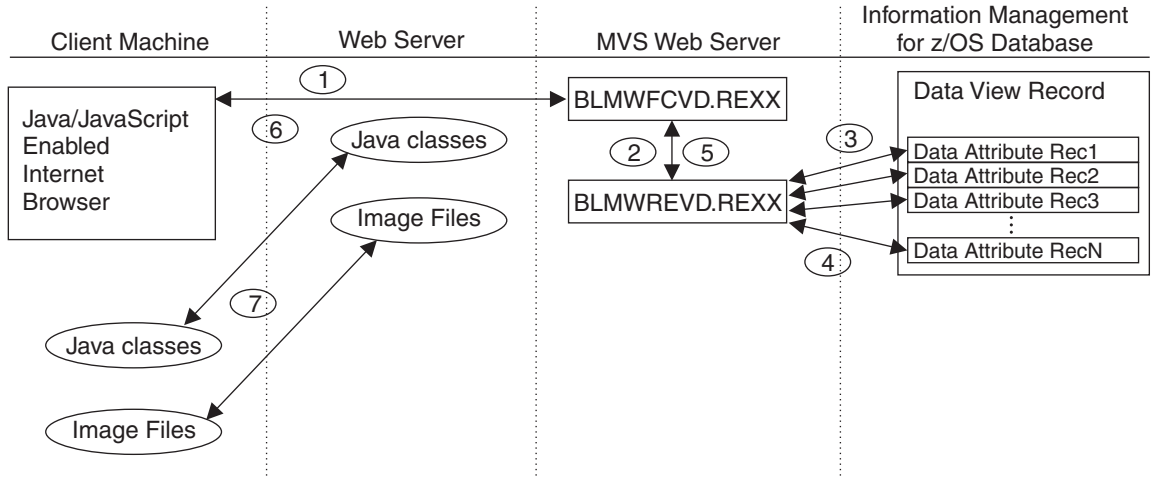


Figure 14. Sample # 2 – Data Field Validation Using Dynamically Generated HTML Forms

The following steps describe the process represented in Figure 14:

1. A user specifies the correct URL to point their Java/JavaScript-enabled Web browser to BLMWFCVD.
2. BLMWFCVD calls BLMWREVD.
3. BLMWREVD performs the Tivoli Information Management for z/OS API calls to retrieve the data view record and the data attribute records contained within the data view record.
4. BLMWREVD retrieves the s-word indexes and validation patterns from each of the attribute records and stores them in global variable stems.
5. After BLMWREVD runs, BLMWFCVD takes the s-word indexes and patterns and generates the HTML form and JavaScripts to communicate to the Java Applets.
6. Once the HTML form and JavaScripts are generated, the MVS server will present the HTML form and load the Java applets.
7. It will also load any image files from a Web server onto the client's browser.

Sample #3—Data Field Validation: Static HTML Forms

Sample # 3 uses a static HTML form instead of the dynamically generated HTML of Sample #2. The validation information (s-word indexes and patterns from the data model records) are still dynamically extracted from data model records. The static HTML form allows you to pick from the set of data attribute records in your data view record which fields you want to be shown in the HTML form. When writing the static HTML form, you must specify the s-word indexes as the NAME parameter in the form field so that the JavaScripts know which pattern to give to the validation applet. Sample # 3 uses two REXX programs, BLMWREVD (which was also used in Sample # 2) which does the retrieval of s-word indexes and patterns from the data model records, and BLMWFCV1, which uses an "include" line for a static HTML form instead of generating the form. The static HTML form used is BLMWHUCF, the predefined create form.

Advanced Modification of Sample Programs

The sample programs provide an illustration of how the Java applets can be used to verify user input. This section provides some additional detail so that you can tailor these programs to meet your specific needs.

Sample # 1

Open BLMWHCF1 and modify the form fields to include your changes. For example, if you want to add a required field in the form, you would need to include the following line:

<input type="text" name=SWORDINDEXOFFIELD value="YVALIDATION_PATTERN">

Change SWORDINDEXOFFIELD and YVALIDATION_PATTERN to the names of your s-word index and your validation pattern.

Sample # 2

If you want to create an HTML form including all the data attribute records in a specified data view record, you will need to create these attribute records and have them in the data view record. Attribute records must contain an s-word index. For example, if you have a data view record called DVMYSAMP that has several data attribute records, you need to open BLMWREVD and BLMWCRTV; look for the word DVSAMPLE and change it to the name of your data view record, DVMYSAMP.

Sample # 3

Sample # 3 provides the flexibility to choose which fields you want to have on your HTML form. Make certain that the data view record specified in BLMWREVD has attribute records that define s-word indexes and validation patterns of the fields you want to validate. Attribute records must contain an s-word index. Also make certain that you use the correct data view record name in BLMWCRTV to do the create process. Open the BLMWHUCF file and add or modify your input fields. For this sample, you only need to specify the s-word indexes as the NAME parameter of the form field.

11

REXX Web connector for OS/390 -- Overview

Overview of the REXX Web connector for OS/390

The REXX Web connector for OS/390 enables you to access a Tivoli Information Management for z/OS database using a Web browser as a client. It is similar to the REXX Web connector for MVS. However, the REXX Web connector for OS/390, unlike the REXX Web connector for MVS, uses the IBM HTTP Server for OS/390. The OS/390 GWAPI interface is used for processing Web browser transactions.

The REXX Web connector for OS/390 uses the IBM HTTP Server for OS/390. It also uses the Tivoli Information Management for z/OS HLAPI/REXX interface. It relies on the IBM HTTP Server for OS/390 GWAPI REXX to establish a REXX environment and to invoke the REXX EXECs in the DGA.

The REXX Web connector for OS/390 is an IBM HTTP Server for OS/390. It is made up of several different elements:

- The client browser
- IBM HTTP Server for OS/390
- The Tivoli Information Management for z/OS HLAPI/REXX interface
- The Database Gateway Application

IBM HTTP Server for OS/390

The IBM HTTP Server for OS/390 provides the interface between the client browser and the Database Gateway Application. It also provides the GWAPI REXX functions. For additional detail on GWAPI REXX, refer to the *HTTP Server Planning, Installation, and Using V5.2 for OS/390*.

Tivoli Information Management for z/OS HLAPI/REXX Interface

Any feature or function available through the HLAPI/REXX interface is available to the Database Gateway Application (DGA). The *Tivoli Information Management for z/OS Application Program Interface Guide* provides detailed information on the HLAPI/REXX interface.

DGA Considerations

The Database Gateway Application contains program logic designed for using the Tivoli Information Management for z/OS HLAPI/REXX Interface and for generating dynamic HTML either from Tivoli Information Management for z/OS records or from search results lists. The sample DGA, using the Tivoli Information Management for z/OS HLAPI/REXX Interface, demonstrates the basic techniques needed to communicate with Tivoli Information Management for z/OS, and also demonstrates a way of interacting with a Web client. The DGA relies on the functions provided by GWAPI REXX. The DGA is described in greater detail in "Modifying the Database Gateway Application" on page 47.

Processing a Request from a Client Browser

When a request from a client browser is sent, the following actions occur:

1. The HTTP Server invokes the GWAPI REXX DLL which establishes a REXX environment.
2. The GWAPI REXX calls the Database Gateway Application to process the client browser transaction.
   - The HLAPI/REXX Interface is called for the requested Tivoli Information Management for z/OS transaction.
   - HTML statements, along with any requested Tivoli Information Management for z/OS data, are built by calling GWAPI REXX service routines.
3. The Database Gateway Application returns to the HTTP Server.
4. At this point the client browser's transaction is complete.

Debugging REXX EXECs

The log file is a combination of the directories specified by the http-errors log file. (Refer to your IBM HTTP Server documentation to determine this file location.) This log file captures all of the REXX environment standard output information (for example, TRACE and SAY statements), including any REXX interpreter output. Once REXX EXECs have been debugged, all logging should be turned off before trying to multitask.

Multitasking

When running the REXX Web connector for OS/390 with the REXX/HLAPI interface, the REXX EXEC BLMWSMLT is provided to allow multitasking. Without it, only one browser request at a time could process a HLAPI/REXX transaction. The DGA as provided is written to work with the HLAPI REXX interface.

Prerequisites

To run the REXX Web connector for OS/390, you must have:

- A Web browser

12

REXX Web connector for OS/390 -- Installation

In order for the IBM HTTP Server to run GWAPI REXX programs, the OS/390 GWAPI REXX software must be installed. After GWAPI REXX is installed, you will need to:

- Create a new service directive in your IBM HTTP Server configuration file on OS/390 UNIX® System Services for the GWAPI REXX DLL, IMWX00. For example:

  ```
  Service /INFOWEB/*.REXX /InfoMgt/GWAPI/bin/IMWX00.so:IMWX00/InfoMgt/GWAPI/bin/BLMWSWRT/*.REXX
  ```

  Note: The IBM HTTP Server configuration file is specified in the –r parameter on the ICSPARM parameter in your HTTP Server startup procedure. If the –r parameter is not specified on ICSPARM or if ICSPARM is not specified, then the configuration file defaults to /etc/httpd.conf. If you specify the –r parameter, but specify a fully-qualified file name, then the path for the configuration file defaults to the /etc directory.

- Add directives for accessing HTML files after the service directive added above. For example:

  ```
  Pass   /INFOWEB/*   /InfoMgt/GWAPI/html/*
  ```

- Modify the PATH statement for the IBM HTTP Server to specify where to find REXX EXECs. The PATH statement is in the **/etc/httpd.envvars** file. This is an example of what you might use:

  ```
  PATH=/bin:.:/usr/lpp/internet/bin:/InfoMgt/GWAPI/bin
  ```

  Note: The IBM HTTP Server configuration file is specified in the –r parameter on the LEPARM parameter in your HTTP Server startup procedure. If the –r parameter is not specified on LEPARM or if LEPARM is not specified, then the configuration file defaults to /etc/httpd.envvars. If you specify the –r parameter, but specify a fully-qualified file name, then the path for the configuration file defaults to the /etc directory.

- You will also need to make sure that the following entries exist for MIME Types located in your configuration file.

  - `Addtype .html text/html`
  - `Addtype .html text/x-ssi-html`

  Alternately, you can also confirm that these entries exist for MIME Types under the Configuration & Administration forms page of your IBM HTTP Server.

Installing the Database Gateway Application

The Database Gateway Application (DGA) consists of REXX EXECs and HTML files. The REXX EXECs and HTML files need to be placed in HFS directories under OS/390 UNIX System Services. The REXX EXECs needed for the DGA are the same ones used for the REXX Web Connector for MVS. They have been written to run when used with either the REXX Web Connector for MVS or the REXX Web Connector for OS/390. The REXX EXECs and HTML files need to be copied from data sets to file directories under OS/390 UNIX System Services.

Note: You may need to substitute the data set names used at your installation for the BLM.SBLMxxxx data set names given in the steps below.

1. Create a directory where the REXX GWAPI EXECs will reside:
   ```
   /InfoMgt/GWAPI/bin
   ```
2. Create a directory where the HTML will reside:
   ```
   /InfoMgt/GWAPI/html
   ```
3. Create an "external link" to the GWAPI REXX. From an OS/390 UNIX System Services session, enter
   ```
   ln –e IMWX00 /InfoMgt/GWAPI/bin/IMWX00.so
   ```
4. Go to your REXX GWAPI directory (created in Step 1) and copy all of the members from data set **BLM.SBLMREXX**. A convenient way to do this is to use the ISPF ISHELL utility. All REXX EXECs in your REXX GWAPI directory should be in upper case.
5. Go to your HTML files directory (created in Step 2) and copy all of the members from data set **BLM.SBLMHTMV**. Provide all of the members with an extension of .html; the member file names should be uppercase, and the file extension .html should be lowercase.
6. Copy **BLMWHELG.html**, **BLMWHPLG.html**, **BLMWHCPR.html**, **BLMWHCIN.html**, and **BLMWHUCF.html** into the directory where IMWX00 resides.
7. Edit the **BLMWSFIN** file in your REXX directory and change the **APPLICATION_ID**, **SESSION_MEMBER**, and **PRIVILEGE_CLASS** as needed. The *Tivoli Information Management for z/OS Application Program Interface Guide* contains additional information about control parameter data blocks (PDBs) used by the HLAPI/REXX interface.
8. If starting your IBM HTTP Server as a catalogued procedure, add the Tivoli Information Management for z/OS SBLMMOD1 data set and your session member load library into your STEPLIB concatenation. If starting the IBM HTTP Server from an OS/390 UNIX System Services session, be sure to include all of the above mentioned data sets in its STEPLIB global variable.
9. If in the future you intend to develop a customized web connector application for your user community, it is suggested that you make the following additional changes to your IBM HTTP server procedure.
   - If starting the IBM HTTP Server as a cataloged procedure, you must allocate a DD statement BLGTSX that points to the data set that contains your REXX TSX EXECs as well as the SBLMTSX data set that was shipped with Tivoli Information Management for z/OS (this data set contains the TSXs used by immediate

notification). You can also concatenate multiple data sets to the BLGTSX DD. Create another data set for user modified or created TSXs, and add it to the BLGTSX DD concatenation. Refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* for more information on starting Tivoli Information Management for z/OS and allocating the BLGTSX data set where your TSX REXX EXECs reside.

The DD statement you allocate should look like the following example:

```
//BLGTSX DD DISP=SHR,DSN=user.execs
//       DD DISP=SHR,DSN=BLM.SBLMTSX
```

In order to receive REXX trace output from your TSXs, as well as any panels or error messages that the TSX may encounter, you should also add the following SYSTSPRT DD to your IBM HTTP Server procedure:

```
//SYSTSPRT DD SYSOUT=*
```

In order to debug TSPs and panel problems using the TRACE or FLOW commands or the TSP PRINT command, you should add the following lines to your IBM HTTP Server procedure:

```
//BLGTRACE DD SYSOUT=*
//BLGFLOW DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

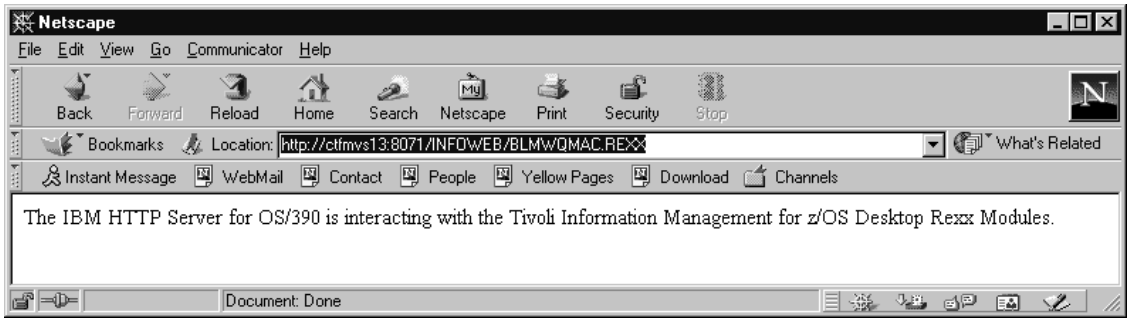
10. Start or restart your IBM HTTP Server. For debugging, start IBM HTTP Server with `-v` or `-vv`.
11. Ensure that the Tivoli Information Management for z/OS BLX Service Provider (BLX-SP) is active.

Verifying HTTP Server File Access

To query whether the connection with the IBM HTTP Server is successful, issue the following command from a browser window:

`http://`*`hostname:portnumber`*`/INFOWEB/BLMWQMAC.REXX`

In the previous example, *hostname* is the host name of your MVS system, and *portnumber* is the port number used to access the IBM HTTP Server for OS/390. If your connection is successful, the following message will result from the command.



Starting the Sample Application

Once you have determined that the connection is successful, you can start the sample application by issuing the following command from a browser window:

`http://`*`hostname:portnumber`*`/INFOWEB/BLMWHDBM.html`

where *hostname* is the host name of your MVS system and *portnumber* is the port number used to access the IBM HTTP Server for OS/390. For more information on using the sample application, see "Sample Database Gateway Application" on page 48.

Migration from REXX Web connector for MVS

If you are currently using REXX Web connector for MVS and would like to migrate to REXX Web connector for OS/390, follow these steps:

1. Copy HTML files to an HFS directory under OS/390 UNIX System Services.
2. Copy REXX EXECs to an HFS directory under OS/390 UNIX System Services.
3. Use the supplied Database Gateway Application as a guide to making changes to your application.
   - To send output back to the browser you will need to change any of your QUEUE statements in REXX to call the IMWXWRT service routine supplied by GWAPI REXX.

     ```
     text = 'Tivoli Information Management for z/OS Search Results List'
     ADDRESS LINKMVS 'IMWXWRT text'
     ```
4. REXX EXECs should also be changed to handle multitasking. Look at the sample DGA for an example of how this can be done. This is the basic flow of how multitasking is implemented:
   a. BLMWSWRT calls BLMWSMLT.
   b. BLMWSMLT finds an available global variable pool and passes it back to BLMWSWRT.
   c. BLMWSWRT then passes the pool to the forms processing routines.
   d. The forms processing routines use the global variable pool to obtain the HLAPI/REXX environment variables.
   e. The forms process routines return to BLMWSWRT, which again calls BLMWSMLT to mark the global variable as available for use.

Note: You may find it easier to make changes to the OS/390 BLMWSWRT rather than migrate the MVS router BLMWSWRT. The changes that you will need to make involve adding callable form processing routines to the router's SELECT statement.

Migration from Earlier Versions of the Web connector for OS/390

If you are migrating from an earlier version of the Web connector for OS/390, follow these steps:

1. Change the return code from BLMWSWRT from **0** to **200**.
2. Records are no longer written to the gwapirx.log file. Records are written to the file specified in the **Error_log** directive in the **httpd.conf** file.

13

REXX Web connector for OS/390 -- Security Considerations

Security is a powerful feature of the Tivoli Information Management for z/OS REXX Web connector for OS/390, not only in the secure functions provided, but also in the ease of use for enabling a secure environment.

The REXX Web connector for OS/390 "inherits" the security features of the HTTP Server server. Configuring security in HTTP Server is usually enough to enable a secure environment, because the REXX Web connector for OS/390 is just another GWAPI program running under HTTP Server. The HTTP Server product is the only server that allows for validating passwords against a Security Authorization Facility (SAF) operating system such as RACF.

It is strongly suggested that you review the HTTP Server security consideration before configuring your HTTP Server environment.

A simple security sample of how you can configure the Web connector follows. This example does not exploit the Secure Sockets Layer (SSL) or Secure Hypertext Transport Protocol (S-HTTP) of the HTTP Server Server.

Sample Security Configuration

Suppose that you want to secure the sample Database Gateway Application with a user password, and further, that you wish to add additional security for the CREATE transaction. You would like the following to occur:

1. When a user accesses the DGA home page BLMWHDBM (or any DGA page) for the first time, they must enter a general user Userid and Password:

   - Userid WEBUSER
   - Password WEBPW

   If these have the correct authentication, the user will be allowed to view the DGA's HTML files and submit SEARCH and RETRIEVE transactions.

2. When a user submits a CREATE transaction for the first time, they must enter a RACF authorized Userid and Password. Assume this to be a logon userid/password.

   You wish to catch CREATE transactions that are entered via the Web browser command line:

   ```
   http://hostname/INFOWEB/CREATE.REXX?S0B59=Doe/Jane...
   ```

   When the CREATE HTML BLMWHCRT is submitted.

The following HTTP Server configuration will enable the above security requirements:

- Create this Protection directive for general users:

```
Protection      WEB_USER {
  ServerID      InfowebUser
  Authtype      Basic
  Passwd file   /infoweb/users/infoweb.pwd
  GetMask       WEBUSER
  }
```

- Create this Protection directive for users with CREATE authority:

```
Protection      WEB_CRT  {
 ServerID      InfowebCrt
 Authtype      Basic
 Passwd file   %%SAF%%
 GetMask       All@(x)
 }
```

- Create an /INFOWEB/USERS directory. In this directory, create a password file INFOWEB.PWD with these definitions:

```
WEBUSER WEBPW:    General_user
```

  You can create a password file with the HTADM command or with HTTP Server configuration panels.

- Create the following Protect directives in this order:

```
Protect   /INFOWEB/CREATE.REXX*  WEB_CRT
Protect   /INFOWEB/*             WEB_USER
```

Restart the HTTP Server server and try out your new security setup by running the Database Gateway Application from a Web browser.

Migration Notes for Security

The Tivoli Information Management for z/OS Web connector for MVS Security routine BLMWSWSE is not carried over to the REXX Web connector for OS/390. Equivalent security is handled by HTTP Server configuration directives. However, you can add additional security routines into the Database Gateway Application.

14

REXX Web connector for OS/2 -- Overview

Overview of the REXX Web connector for OS/2

The REXX Web connector for OS/2 enables you to access a Tivoli Information Management for z/OS database using a Web browser as a client. It is analogous to the REXX Web connector for MVS. However, the REXX Web connector for OS/2 is a stand-alone OS/2 application which implements both the OS/2 IBM Internet Connection Server Version 4.1 for OS/2 WARP® (ICS) and the Tivoli Information Management for z/OS REXX HLAPI/2 client feature.

The combination of ICSS and the Tivoli Information Management for z/OS REXX HLAPI/2 gives the REXX Web connector for OS/2 powerful advantages:

- The REXX Web connector for OS/2 supports both the TCP/IP and APPC/MVS communication protocols to the MVS host. (The Tivoli Information Management for z/OS REXX Web connector for MVS supports only the TCP/IP protocol.)
- Multitasking of Web browser client requests is intrinsic to the REXX Web connector for OS/2. (The Tivoli Information Management for z/OS REXX Web connector for MVS queues Web client requests to a single task.)
- The REXX Web connector for OS/2 provides Internet Connection Service (ICS) Security. The REXX Web connector for OS/2 inherits security features from both the Internet Connection Server and the Internet Connection Secure Server.
- The REXX Web connector for OS/2 provides distributed processing to different Tivoli Information Management for z/OS MVS systems.

The REXX Web connector for OS/2 is an application that runs on an OS/2 machine. It is an ICS ICAPI program that is called when a client Web browser directs a particular transaction to the ICSS server installed on your OS/2 machine. The REXX Web connector for OS/2 has two components:

- The Web connector Server
- The database gateway application

Both of these components are described in “REXX Web connector for OS/2 -- Functional Interface” on page 77.

Processing a Request from a Client Browser

When a request from a client browser is sent, the following actions occur:

1. The ICSS server invokes the Web connector server as an ICAPI application
2. The Web connector server calls the database gateway application to process the client browser transaction:

- The REXX HLAPI/2 interface is called for the requested Tivoli Information Management for z/OS transaction. As with any REXX HLAPI/2 program, an MVS server (MRES or RES) processes the Tivoli Information Management for z/OS request and control is returned to the REXX HLAPI/2 interface.
- HTML statements, along with any requested Tivoli Information Management for z/OS data, are built by calling Web connector service routines.

3. The database gateway application returns to the Web connector server, which in turn returns to ICSS.
4. At this point, the client browser's transaction is complete.

Prerequisites

To run the Tivoli Information Management for z/OS REXX Web connector for OS/2, you must have:

- A Web browser
- OS/2 WARP Version 4
- Lotus® Domino™ for OS/2
- HLAPI/2

Installation

Install REXX Web connector for OS/2 by following these steps:

Note: You can only install Web connector on an HPFS drive.

1. Switch to, or start, an OS/2 window or an OS/2 full screen session.
2. If you already have Web connector installed, delete it by changing to the Web connector installation directory and running EPFINSTS.
3. Insert the Web connector CD-ROM into a CD-ROM drive.
4. Type the following command at the OS/2 command prompt, then press Enter:

   ```
   e:\web\os2\install
   ```

   where:

   e Is the drive letter of the CD-ROM drive that contains the Web connector CD-ROM.
5. Read the information in the instructions window, then select **Continue**.
6. In the Install window, if you want the Installation and Maintenance Utility to update your CONFIG.SYS file, select **OK** and go on to step 7. If you do not select **OK**, changes are put in a file called CONFIG.ADD.

   If you do not want the Installation and Maintenance Utility to update your CONFIG.SYS file, do the following:

   a. De-select **Update CONFIG.SYS** before you select **OK**.

   b. Modify the CONFIG.SYS file manually before you shut down and restart your workstation or start Web connector. Modify the CONFIG.SYS file using the information in the CONFIG.ADD file. It is in the same directory as your

CONFIG.SYS file. This file is not created until the Web connector Installation and Maintenance Utility has completed its part of the installation.

7. In the Install - Directories window:
   a. Select the component you want to install.
   b. Type the target paths in which to install the Web connector files. You can accept the default values or change them. If the paths do not exist, they will be created. The default directory is C:\INFOWEB.

      Note: You can select **Disk space** to determine the amount of available space on the fixed disk drives in your workstation.

   c. Select **Install**

      The Web connector files are transferred from the installation CD-ROM to your workstation. The Install - Progress window indicates progress.
8. When the transfer is complete, a message appears to indicate that Web connector is installed. Select **OK**.

Note: If you update the CONFIG.SYS file during the installation, you must shut down your workstation and start it again before starting Web connector.

The Web connector installation is complete. After you start your workstation again, verify the installation.

15

REXX Web connector for OS/2 -- Functional Interface

BLMWWEBS, the Web connector server, is the ICAPI program that ICSS calls to process Tivoli Information Management for z/OS-type transactions and related services.

BLMWWEBS is an object code DLL that contains the server functions CALLWEB, ENDWEB, and INITWEB.

Initweb

The INITWEB function runs when ICSS is started during its initialization phase. The INITWEB function erases any residual global variables file (USER.INI) produced from an earlier ICSS Web connector. It can be enhanced to do other initialization functions such as setting global session variable. The ICSS process step associated with this function is *Service*.

ICSS knows to call BLMWWEBS if you configured it with this ICAPI Serverinit directive:

```
e:/infoweb/bin/blmwwebs:Initweb
```

Initweb calls the REXX Database Gateway Application initialization routine. Initweb processing:

- Sets up service environment (such as logging for the REXX routines) for the REXX initialization routine
- Calls the Database Gateway Application initialization routine BLMWSINI
- Returns to ICSS

Callweb

The **CALLWEB** function sets up the Web connector environment and calls the Database Gateway Application. The ICSS process step associated with this function is *Service*.

When an HTML form is submitted that matches the directive for an Infoweb CALLWEB function, ICSS gives control to the ICAPI BLMWWEBS service CALLWEB. ICSS knows to call BLMWWEBS if you configured it with this ICAPI:

```
/callweb*  e:\infoweb\bin\blmwwebs:Callweb
```

Callweb is the interface between ICSS and the Database Gateway Application routines. Callweb processing:

- Sets up service environment (such as logging for the REXX routines) in the Database Gateway Application.

- Calls the Database Gateway Application router BLMWSWRT.
- Returns to ICS.

Endweb

The Endweb function runs when ICSS terminates. The Endweb function cleans up Database Gateway Application resources. The ICSS process step associated with this function is *Server Termination*.

ICSS knows to call BLMWWEBS if you configured it with this ICAPI Serverterm directive:

```
e:\infoweb\bin\blmwwebs:Endweb
```

Endweb calls the REXX Database Gateway Application termination routine. Endweb processing:

- Sets up the logging environment file BLMWWEBS.END in the INFOWEB\BIN directory. SAY and TRACE output from BLMWSTRM are logged to this file.
- Calls the Database Gateway Application termination routine BLMWSTRM.
- Returns to ICS.

Callable Service Routines

These service routines are callable from the Database Gateway Application REXX programs. They provide bindings to some of the predefined ICSS functions that your REXX programs cannot normally access. The syntax and description are as follows:

QUEUEIT string
: Write the requested string to the body of the response. An HTTPD_write function is performed.

SET_HTTPD variable_name value
: Sets a variable (HTTP_RESPONSE, for example) with a value. An HTTPD_set function is performed.

GET_HTTPD variable_name
: The value of variable_name (QUERY_STRING, for example) is returned in the REXX variable result. An HTTPD_extract function is performed.

16

REXX Web connector for OS/2 -- Security Considerations

Security Considerations

Security is a powerful feature of the Tivoli Information Management for z/OS REXX Web connector for OS/2, not only in the secure functions provided, but also in the ease of use for enabling a secure environment.

The REXX Web connector for OS/2 “inherits” the security features of both the Internet Connection Server and the Internet Connection Secure Server. Configuring security in ICSS is usually enough to enable a secure environment, because the REXX Web connector for OS/2 is just another ICAPI program running under ICSS.

It is strongly suggested that you review the ICSS security sections in the IBM Internet Connection Server Webmaster’s Guide for OS/2 WARP before configuring your ICSS environment.

A simple security sample of how you can configure the Web connector follows. Note that this sample configuration will run for both the Internet Connection Server and Internet Connection Secure Server. However, it does not exploit the Secure Sockets Layer (SSL) or Secure Hypertext Transport Protocol (S-HTTP) of the Internet Connection Secure Server.

Sample Security Configuration

Suppose that you want to secure the sample Database Gateway Application with a user password, and further, that you wish to add additional security for the CREATE transaction. You would like the following to occur:

1. When a user accesses the DGA home page BLMWHDBM (or any DGA page) for the first time, they must enter a general user Userid and Password:

   - Userid WEBUSER
   - Password WEBPW

   If these have the correct authentication, the user will be allowed to view the DGA’s HTML files and submit SEARCH and RETRIEVE transactions.

2. When a user submits a CREATE transaction for the first time, they must enter a RACF authorized Userid and Password. These must be one of the REXX HLAPI/2 session Security_ID and Passwords pairs:

   - Userid SAMPID Password PASSWORD
   - Userid SAMPID2 Password PASSWRD2

You also wish to catch CREATE transactions that are entered via the Web browser command line:

```
http://hostname/callweb/CREATE.REXX?S0B59=Doe/Jane...
```

In this case, the create HTML BLMWHCRT would be bypassed.

The following ICSS configuration will enable the above security requirements:

- Create this Protection directive for general users:

```
Protection    WEB_USER {
  ServerID    InfowebUser
  Authtype    Basic
  Passwd      E:\infoweb\users\infoweb.pwd
  GetMask     WEBUSER,SAMPID,SAMPID2
  }
```

- Create this Protection directive for users with CREATE authority:

```
Protection    WEB_CRT  {
 ServerID    InfowebCrt
 Authtype    Basic
 Passwd      E:\infoweb\users\infoweb.pwd
 GetMask     SAMPID,SAMPID2
 }
```

- Create an E:\INFOWEB\USERS directory. In this directory, create a password file INFOWEB.PWD with these definitions:

```
WEBUSER WEBPW:    General_user
SAMPID  PASSWORD: Priv_user
SAMPID2 PASSWRD2: Priv_user
```

  You can create a password file with the HTADM command or with ICSS configuration panels.

- Create the following Protect directives:

```
Protect   /infoweb/*             WEB_USER
Protect   /callweb/CREATE.REXX*  WEB_CRT
```

Restart the ICSS server and try out your new security setup by running the Database Gateway Application from a Web browser.

Migration Notes for Security

The Tivoli Information Management for z/OS Web connector for MVS Security routine BLMWSWSE is not carried over to the REXX Web connector for OS/2. Equivalent security is handled by ICSS configuration directives. However, you can add additional security routines into the Database Gateway Application.

17

REXX Web connector for OS/2 -- Database Gateway Application

The Database Gateway Application (DGA)

The Database Gateway Application (DGA) consists of REXX programs that provide gateway connectivity between HTML forms on a Web browser and Tivoli Information Management for z/OS through the REXX HLAPI/2 client. The entire Database Gateway Application can be modified or enhanced to fit your needs.

DGA Service Routines

DGA Router - BLMWSWRT

The routine BLMWSWRT is called when an HTML form is sent from a remote browser and is received by the Web connector server routine Callweb. BLMWSWRT routes the request to a Forms Processing Routine. While the routine name (BLMWSWRT) and called interface must always remain intact, you can modify the remainder of the processing routine names.

BLMWSWRT Interface

No parameters are passed to BLMWSWRT.

BLMWSWRT Operation

The URL passed is mapped to an actual forms processing routine and that routine is called by the router. The forms processing routine should return any error condition by returning data through the following REXX statement:

```
RETURN code message
```

Where

code A valid HTTP return code.

message A message string or null.

DGA Initialization — BLMWSINI

This routine is called during ICSS server initialization. Upon initialization, the Web connector server routine Initweb calls BLMWSINI to erase any residual global variable file named USER.INI. No parameters are passed to BLMWSINI. The return statement is always of the form:

```
RETURN 0
```

DGA Termination - BLMWSTRM

This routine is called during ICSS server termination. Upon termination, the Web connector server routine Endweb calls BLMWSTRM to shut down active REXX HLAPI/2 sessions and clean up global pool variables. No parameters are passed to BLMWSTRM. The return statement is always of the form:

```
RETURN 0
```

DGA Global Variable Pool Service - BLMWSMLT

This routine provides the following services:

- Get a global variable pool and return its name to the caller.
- Free a requested global variable pool.

BLMWSMLT Interface

An example of using a GET request with BLMWSMLT is:

```
CALL BLMWSMLT 'GET'
```

For a GET request, the return statement is always of the form:

```
RETURN  0  name
         or
RETURN 0  error_message
```

An example of using a FREE request with BLMWSMLT is:

```
CALL BLMWSMLT 'FREE' poolele
```

where poolele is the name of a global variable pool name to free.

For a FREE request the return statement is always of the form:

```
RETURN  0
```

DGA Forms Processing Routines

The Database Gateway Application provides the following sample forms processing routines to illustrate how an application can be constructed:

BLMWFCRT	Create record routine
BLMWFRVW	View record routine
BLMWFSCH	Search record routine

The forms processing routines are essentially the same as the sample Tivoli Information Management for z/OS Web connector for MVS routines. For an explanation of these routines, refer to “Sample DGA REXX Forms Service Routines” on page 46.

The differences between the OS/2 and MVS forms processing routines are explained in “Migrating Forms Processing Routines” on page 84.

DGA Forms Service Routines

DGA Forms Initialization Service - BLMWSFIN

The Database Gateway Application provides BLMWSFIN to initialize REXX HLAPI/2 sessions and create REXX global variables for the session. BLMWSFIN is essentially the

same as the sample REXX Web connector for MVS routine. Differences between the OS/2 and MVS BLMWSFIN routines are explained in “Migrating the BLMWSFIN Routine” on page 84.

General Migration Notes for MVS Database Gateway Application to OS/2

The following changes apply to all DGA REXX routines in the Tivoli Information Management for z/OS Web connector for MVS.

- MVS RGV (REXX Global Variable) Service calls are replaced by OS/2 SYSINI services. Refer to “Migration Notes for Global Variables” on page 88 for additional information.
- Change QUEUE statements to QUEUEIT calls. The REXX Web connector for MVS uses the REXX QUEUE statement to send HTML output to the Web server. In the REXX Web connector for OS/2, a callable Web server routine, QUEUEIT is provided, which sends data to the client browser.
- RXFUNCADD statements must be added for each external C service call. In this example, from the router BLMWSWRT, QUEUEIT is invoked:

  ```
  Call RxFuncAdd 'QUEUEIT',blmwwebs,'QUEUEIT'
  ```

  This command has the effect of “registering” the command. Any subsequent invocations of the QUEUEIT command do not need to be preceded by the RXFUNCADD statement.
- OS/2 uses ASCII translations, MVS uses EBCDIC translations. For example, you must use the EBCDIC equivalent of the carriage return line feed (crlf) command if you are using the Web connector for MVS, or the ASCII equivalent of the carriage return line feed crlf command if you are using the REXX Web connector for OS/2. The MVS crlf variable definition is :

  ```
  crlf = x2c('0D25');           /* EBCDIC equivalent of ASCII CRLF */
  ```

  The OS/2 crlf variable definition is:

  ```
  crlf = x2c('0D0A');           /* ASCII CRLF                      */
  ```
- REXX HLAPI/2 calls replace MVS HLAPI REXX invocations. The OS/2 DGA termination routine BLMWSTRM replaces the MVS forms termination routine BLMWSFTE.
- Deletion of Web connector for MVS Server Routines. These routines, used in the Web connector for MVS Server, are no longer needed in the REXX Web connector for OS/2:
  - BLMWSWGM
  - BLMWSWAS
  - BLMWSWPA
  - BLMWSWSE
  - BLMWSW64

Specific Migration Notes for MVS Database Gateway Application to OS/2

You will need to consider the following factors when migrating from the MVS Database Gateway Application to the OS/2 Database Gateway Application.

- The OS/2 BLMWSWRT routine retrieves its own environment variables. No arguments are passed to BLMWSWRT. The Web server routine GET_HTTPD is invoked to retrieve environment variables. (In the Web connector for MVS server, arguments were passed.)
- The OS/2 BLMWSWRT routine sets HTTP header fields. For the REXX Web connector for OS/2, the BLMWSWRT routine calls the Web server routine SET_HTTPD to set fields such as HTTP_RESPONSE in the HTTP header. (In the Web connector for MVS server, BLMWWEBS provided this function.)
- A global variable pool name is obtained and passed to a called form processing routine. The DGA service BLMWSMLT is called to obtain and release this global variable pool name.
- The OS/2 BLMWSWRT routine post parses the URL "body". Special characters such as + are translated to blanks and the & delimiters are replaced by X'FF'. (In the Web connector for MVS server, BLMWWEBS and services function BLMWSWPA provided this function.)
- The forms processing routine name *member* is obtained with this parse statement:

  ```
  Parse upper var pi 'CALLWEB/'member'.'
  ```

- The OS/2 BLMWSWRT routine builds error responses sent to the client browser. (In the Web connector for MVS server, BLMWWEBS provided this function.)

Migrating the BLMWSFIN Routine

- You must add an RxfuncAdd call to register the REXX HLAPI/2 interface.
- The OS/2 BLMWSFIN routine can start multiple sessions to enable multiprocessing of transactions.

Migrating Forms Processing Routines

- The OS/2 forms processing routine input interface has changed:

  ```
  /* One parameter passed */
  parse arg global
  ```

Note that you can also add other parameters, such as the query_string body, as long as the call is changed from the router BLMWSWRT also.

Communication Protocols

The Tivoli Information Management for z/OS REXX Web connector for OS/2 supports two communication protocols between OS/2 and the connected MVS system where the Tivoli Information Management for z/OS database resides. These protocols are:

- TCP/IP
- APPC

The choice of protocol is controlled in the HLAPI/2 database profile. The name of the HLAPI/2 database profile used by the Tivoli Information Management for z/OS OS/2 Web connector is defined in the Database Gateway Application routine BLMWSFIN in field DATABASE_PROFILE. (If APPC is used, you do not need to have TCP/IP on MVS.)

Migration Notes for Communication Protocol

The Tivoli Information Management for z/OS Web connector for MVS supports only the TCP/IP communication protocol.

Multithreaded Transactions

The REXX Web connector for OS/2 allows for concurrent processing of multiple Web browser transactions. This asynchronous processing is feasible because ICSS supports multiple ICAPI programs running simultaneously in its process. Because the REXX Web connector for OS/2 is an ICAPI program, much of the multithreading work is done without the Web connector needing to do anything.

However, this does not cover concurrent REXX HLAPI/2 sessions. If the same security_id and password are used in separate sessions, and the Web connector is concurrently processing transactions for these sessions, the HLAPI/2 client will reuse the same conversation with the MVS MRES. Ultimately, the HLAPI/2 client will queue the transactions at the conversation level. This is true for both the APPC and TCP/IP communication protocols.

To solve this queueing problem and enable true multithreading, unique security_id and password pairs are dedicated for each session. This is done in the DGA REXX forms Initialization Service BLMWSFIN. The extract below allows for two true concurrent sessions. Additional concurrent sessions will queue on the first session.

```
Select;
  When global = 'free.1' then
   do;
      security_id = 'SAMPID';
      password    = 'PASSWORD';
   end;
  When global = 'free.2' then
   do;
      security_id = 'SAMPID2';
      password    = 'PASSWRD2';
   end;
  Otherwise
   do;
      security_id = 'SAMPID';
      password    = 'PASSWORD';
   end;
 End;
```

You can increase the number of concurrent sessions by adding additional security_id and password pairs to the above Select statement.

Migration Notes for Multithreaded Transactions

The REXX Web connector for MVS does not support multithreading of browser transactions. Simultaneous transactions are queued, which degrades the response time at the requesting client browser.

18

REXX Web connector for OS/2 -- Global Variables

Global Variables

Global variables are used by the Database Gateway Application in two different ways:

- To store REXX variables that are used between Web browser transactions. This is known as *state* data. Its most common use is to store the REXX HLAPI/2 environment variable BLG_ENVP for the entire ICS session. This method is used extensively in the Database Gateway Application shipped with the REXX Web connector for OS/2.
- To store REXX variables for only the duration of the Web browser transaction. This is a convenient way of storing and retrieving variables between REXX routines without passing the variables as parameters.

The Tivoli Information Management for z/OS REXX Web connector for OS/2 uses the service SYSINI to control global variables. SYSINI is part of the OS/2 RexxUtil package.

SysIni Usage

The following examples show how the DGA uses SYSINI. To store the variable BLG_ENVP, use the syntax:

```
userf = '\infoweb\bin\user.ini'
result = SysIni(userf,global,'BLG_ENVP',BLG_ENVP)
```

Storing REXX variables is used in the forms Initialization routine BLMWSFIN.

To retrieve the variable BLG_ENVP, use the syntax:

```
result = SysIni(userf,global,BLG_ENVP)
BLG_ENVP = result
```

Retrieving REXX variables is used throughout the forms processing routines.

To retrieve all the names of all the global variable pools and have them stored in the stem variable MULT use the syntax:

```
userf = '\infoweb\bin\user.ini'
result = SysIni(userf,mults,'ALL:','mult')
```

Retrieving global variable pool names is used in the Global Variable Pool Service BLMWSMLT.

To delete all the global pool variable names, use the syntax:

```
userf = '\infoweb\bin\user.ini'
result = SysIni(userf,mults,'DELETE:')
```

Deletion of the global variable pool names is done in the Termination routine BLMWSTRM.

Migration Notes for Global Variables

- Change all MVS RGV (REXX Global Variable) Service calls to OS/2 SysIni services. The following examples show use of MVS RGV calls and the replacement OS/2 SysIni invocations.
  - MVS GET global variable sample called by forms processing routines

```
result = blmxvgt('GLOBAL','JOBSTEP','blg*')
parse var result result result_text
if result > 8 then return '400' 'Global variable error.' result_text;
if result = 8 then do;
  call BLMWSFIN;    /* API not initialized */
  parse var result result result_text
  if result > 0 then return '502' 'Initialization error ' result_text;
  result = blmxvgt('GLOBAL','JOBSTEP','blg*')
  parse var result result result_text
  if result > 0 then return '502' 'Initialization error ' result_text;
  end;
```

  - OS/2 GET global variable sample called by forms processing routines

```
sys_result = SysIni(userf,global,BLG_ENVP)

/*  Check for BLG_ENVP being initialized  */
If sys_result = 'ERROR:' then do;
  call BLMWSFIN global;    /* API not initialized */

  parse var result result result_text
  if result > 0 then
   do
    return  '502 Initialization error ' result_text;
   end
  result = SysIni(userf,global,BLG_ENVP)
  parse var result sys_result .
  if sys_result = 'ERROR:' then
   do
    return '502   Initialization error: Global variable BLG_ENV'
   end
  end;

BLG_ENVP = sys_result;
```

  - MVS PUT global variable sample called by initialization service BLMWSFIN.

```
result = blmxvpt('GLOBAL','JOBSTEP','blg*')
parse var result result result_text
if result > 4 then return '500' 'Global variables error.' result_text;
```

  - OS/2 PUT global variable sample called by initialization service BLMWSFIN.

```
result = SysIni(userf,global,'BLG_ENVP',BLG_ENVP)
parse var result result result_text
if result = 'ERROR:' then
  return '502 Initialization error: Global variable BLG_ENVP
```

- Other Global Variable Changes and Differences
  - The OS/2 global variable pool name is now a variable passed as an argument from the router BLMWSWRT. (In the MVS DGA it was hardcoded as GLOBAL.)
    - MVS DGA

```
result = blmxvgt('GLOBAL','JOBSTEP','blg*')
```

    - OS/2 DGA

```
parse arg body , global;
            .
            .
result = SysIni(userf,global,BLG_ENVP)
```

- The OS/2 SYSINI sample does not initialize or terminate environments as in the MVS implementation. In the MVS implementation, you may have calls such as these:

```
/** Initialize RGV **/
result = BLMXSMK('GLOBAL','JOBSTEP')

/** Drop the existing global variables                              */
result = blmxvdr('GLOBAL','JOBSTEP','blg*')
```

There are no equivalent OS/2 SYSINI calls to these MVS RGV code statements.

19

REXX Web Connector for OS/2 -- Logging

The Tivoli Information Management for z/OS REXX Web connector for OS/2 logs STDOUT and STDERR information to the ICSS error log, which by the default setting is displayed at the ICSS console. The Database Gateway Application contains two logging methods:

- Using REXX SAY statements:

```
/* Add a say statement in routine BLMWSWRT */

say 'In the router routine BLMWSWRT'
```

- Using REXX TRACE statements:

```
/* Add a trace on result statement in  routine BLMWSWRT */

Trace 'R'
```

Restriction: Logging for simultaneous transactions (multithreaded) currently does not work. This is being investigated.

Migration Notes for Logging

The REXX Web connector for MVS has a set of log codes issued by its server BLMWWEBS. These notes were not migrated to the REXX Web connector for OS/2, because most of the processing which issued log codes in the REXX Web connector for MVS is now handled in the ICSS layer on OS/2.

A

Relating Publications to Specific Tasks

Your data processing organization can have many different users performing many different tasks. The books in the Tivoli Information Management for z/OS library contain task-oriented scenarios to teach users how to perform the duties specific to their jobs.

The following table describes the typical tasks in a data processing organization and identifies the Tivoli Information Management for z/OS publication that supports those tasks. See "The Tivoli Information Management for z/OS Library" on page 99 for more information about each book.

Typical Tasks

Table 1. Relating Publications to Specific Tasks

If You Are:	And You Do This:	Read This:
Planning to Use Tivoli Information Management for z/OS	Identify the hardware and software requirements of Tivoli Information Management for z/OS. Identify the prerequisite and corequisite products. Plan and implement a test system.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
Installing Tivoli Information Management for z/OS	Install Tivoli Information Management for z/OS. Define and initialize data sets. Create session-parameters members.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference* *Tivoli Information Management for z/OS Integration Facility Guide*
	Define and create multiple Tivoli Information Management for z/OS BLX-SPs.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
	Define and create APPC transaction programs for clients.	*Tivoli Information Management for z/OS Client Installation and User's Guide*
	Define coupling facility structures for sysplex data sharing.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
Diagnosing problems	Diagnose problems encountered while using Tivoli Information Management for z/OS	*Tivoli Information Management for z/OS Diagnosis Guide*

Table 1. Relating Publications to Specific Tasks (continued)

If You Are:	And You Do This:	Read This:
Administering Tivoli Information Management for z/OS	Manage user profiles and passwords. Define and maintain privilege class records. Define and maintain rules records.	*Tivoli Information Management for z/OS Program Administration Guide and Reference* *Tivoli Information Management for z/OS Integration Facility Guide*
	Define and maintain USERS record. Define and maintain ALIAS record. Implement GUI interface. Define and maintain command aliases and authorizations.	*Tivoli Information Management for z/OS Program Administration Guide and Reference*
	Implement and administer Notification Management. Create user-defined line commands. Define logical database partitioning.	*Tivoli Information Management for z/OS Program Administration Guide and Reference*
	Create or modify GUI workstation applications that can interact with Tivoli Information Management for z/OS. Install the Tivoli Information Management for z/OS Desktop on user workstations.	*Tivoli Information Management for z/OS Desktop User's Guide*
Maintaining Tivoli Information Management for z/OS	Set up access to the data sets. Maintain the databases. Define and maintain privilege class records.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference* *Tivoli Information Management for z/OS Program Administration Guide and Reference*
	Define and maintain the BLX-SP. Run the utility programs.	*Tivoli Information Management for z/OS Operation and Maintenance Reference*
Programming applications	Use the application program interfaces.	*Tivoli Information Management for z/OS Application Program Interface Guide*
	Use the application program interfaces for Tivoli Information Management for z/OS clients.	*Tivoli Information Management for z/OS Client Installation and User's Guide*
	Create Web applications using or accessing Tivoli Information Management for z/OS data.	*Tivoli Information Management for z/OS World Wide Web Interface Guide*

Table 1. Relating Publications to Specific Tasks (continued)

If You Are:	And You Do This:	Read This:
Customizing Tivoli Information Management for z/OS	Design and implement a Change Management system. Design and implement a Configuration Management system. Design and implement a Problem Management system.	*Tivoli Information Management for z/OS Problem, Change, and Configuration Management*
	Design, create, and test terminal simulator panels or terminal simulator EXECs. Customize panels and panel flow.	*Tivoli Information Management for z/OS Terminal Simulator Guide and Reference* *Tivoli Information Management for z/OS Panel Modification Facility Guide*
	Design, create, and test Tivoli Information Management for z/OS formatted reports.	*Tivoli Information Management for z/OS Data Reporting User's Guide*
	Create a bridge between NetView® and Tivoli Information Management for z/OS applications. Integrate Tivoli Information Management for z/OS with Tivoli distributed products.	*Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications*
Assisting Users	Create, search, update, and close change, configuration, or problem records. Browse or print Change, Configuration, or Problem Management reports.	*Tivoli Information Management for z/OS Problem, Change, and Configuration Management*
	Use the Tivoli Information Management for z/OS Integration Facility.	*Tivoli Information Management for z/OS Integration Facility Guide*
Using Tivoli Information Management for z/OS	Learn about the Tivoli Information Management for z/OS panel types, record types, and commands. Change a user profile.	*Tivoli Information Management for z/OS User's Guide*
	Learn about Problem, Change, and Configuration Management records.	*Tivoli Information Management for z/OS Problem, Change, and Configuration Management*
	Receive and respond to Tivoli Information Management for z/OS messages.	*Tivoli Information Management for z/OS Messages and Codes*
	Design and create reports.	*Tivoli Information Management for z/OS Data Reporting User's Guide*

B

Tivoli Information Management for z/OS Courses

Education Offerings

Tivoli Information Management for z/OS classes are available in the United States and in the United Kingdom. For information about classes outside the U.S. and U.K., contact your local IBM representative or visit **http://www.training.ibm.com** on the World Wide Web.

United States

IBM Education classes can help your users and administrators learn how to get the most out of Tivoli Information Management for z/OS. IBM Education classes are offered in many locations in the United States and at your own company location.

For a current schedule of available classes or to enroll, call 1-800-IBM TEACh (1-800-426-8322). On the World Wide Web, visit:

http://www.training.ibm.com

to see the latest course offerings.

United Kingdom

In Europe, the following public courses are held in IBM's central London education centre at the South Bank at regular intervals. On-site courses can also be arranged.

For course schedules and to enroll, call Enrollments Administration on 0345 581329, or send an e-mail note to:

contact_educ_uk@vnet.ibm.com

On the World Wide Web, visit:

http://www.europe.ibm.com/education-uk

to see the latest course offerings.

C

Where to Find More Information

The Tivoli Information Management for z/OS library is an integral part of Tivoli Information Management for z/OS. The books are written with particular audiences in mind. Each book covers specific tasks.

The Tivoli Information Management for z/OS Library

The publications shipped automatically with each Tivoli Information Management for z/OS Version 7.1 licensed program are:

- *Tivoli Information Management for z/OS Application Program Interface Guide*
- *Tivoli Information Management for z/OS Client Installation and User's Guide **
- *Tivoli Information Management for z/OS Data Reporting User's Guide **
- *Tivoli Information Management for z/OS Desktop User's Guide*
- *Tivoli Information Management for z/OS Diagnosis Guide **
- *Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications **
- *Tivoli Information Management for z/OS Integration Facility Guide **
- *Tivoli Information Management for z/OS Licensed Program Specification*
- *Tivoli Information Management for z/OS Master Index, Glossary, and Bibliography*
- *Tivoli Information Management for z/OS Messages and Codes*
- *Tivoli Information Management for z/OS Operation and Maintenance Reference*
- *Tivoli Information Management for z/OS Panel Modification Facility Guide*
- *Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
- *Tivoli Information Management for z/OS Program Administration Guide and Reference*
- *Tivoli Information Management for z/OS Problem, Change, and Configuration Management**
- *Tivoli Information Management for z/OS Reference Summary*
- *Tivoli Information Management for z/OS Terminal Simulator Guide and Reference*
- *Tivoli Information Management for z/OS User's Guide*
- *Tivoli Information Management for z/OS World Wide Web Interface Guide*

Note: Publications marked with an asterisk (*) are shipped in softcopy format only.

Also included is the Product Kit, which includes the complete online library on CD-ROM.

To order a set of publications, specify order number SBOF-7028-00.

Additional copies of these items are available for a fee.

Publications can be requested from your Tivoli or IBM representative or the branch office serving your location. Or, in the U.S., you can call the IBM Publications order line directly by dialing 1-800-879-2755.

The following descriptions summarize all the books in the Tivoli Information Management for z/OS library.

Tivoli Information Management for z/OS Application Program Interface Guide, SC31-8737-00, explains how to use the low-level API, the high-level API, and the REXX interface to the high-level API. This book is written for application and system programmers who write applications that use these program interfaces.

Tivoli Information Management for z/OS Client Installation and User's Guide, SC31-8738-00, describes and illustrates the setup and use of Tivoli Information Management for z/OS's remote clients. This book shows you how to use Tivoli Information Management for z/OS functions in the AIX®, CICS®, HP-UX, OS/2, Sun Solaris, Windows NT®, and OS/390 UNIX System Services environments. Also included in this book is complete information about using the Tivoli Information Management for z/OS servers.

Tivoli Information Management for z/OS Data Reporting User's Guide, SC31-8739-00, describes various methods available to produce reports using Tivoli Information Management for z/OS data. It describes Tivoli Decision Support for Information Management (a Discovery Guide for Tivoli Decision Support), the Open Database Connectivity (ODBC) Driver for Tivoli Information Management for z/OS, and the Report Format Facility. A description of how to use the Report Format Facility to modify the standard reports provided with Tivoli Information Management for z/OS is provided. The book also illustrates the syntax of report format tables (RFTs) used to define the output from the Tivoli Information Management for z/OS REPORT and PRINT commands. It also includes several examples of modified RFTs.

Tivoli Information Management for z/OS Desktop User's Guide, SC31-8740-00, describes how to install and use the sample application provided with the Tivoli Information Management for z/OS Desktop. The Tivoli Information Management for z/OS Desktop is a Java-based graphical user interface for Tivoli Information Management for z/OS. Information on how to set up data model records to support the interface and instructions on using the Desktop Toolkit to develop your own Desktop application are also provided.

Tivoli Information Management for z/OS Diagnosis Guide, GC31-8741-00, explains how to identify a problem, analyze its symptoms, and resolve it. This book includes tools and information that are helpful in solving problems you might encounter when you use Tivoli Information Management for z/OS.

Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications, SC31-8744-00, describes the steps to follow to make an automatic connection between NetView and Tivoli Information Management for z/OS applications. It also explains how to customize the application interface which serves as an application enabler for the NetView Bridge and discusses the Tivoli Information Management for z/OS NetView AutoBridge. Information on interfacing Tivoli Information Management for z/OS with other Tivoli management software products or components is provided for Tivoli Enterprise Console, Tivoli Global Enterprise Manager, Tivoli Inventory, Tivoli Problem Management, Tivoli Software Distribution, and Problem Service.

Tivoli Information Management for z/OS Integration Facility Guide, SC31-8745-00, explains the concepts and structure of the Integration Facility. The Integration Facility provides a task-oriented interface to Tivoli Information Management for z/OS that makes the

Tivoli Information Management for z/OS applications easier to use. This book also explains how to use the panels and panel flows in your change and problem management system.

Tivoli Information Management for z/OS Master Index, Glossary, and Bibliography, SC31-8747-00, combines the indexes from each hardcopy book in the Tivoli Information Management for z/OS library for Version 7.1. Also included is a complete glossary and bibliography for the product.

Tivoli Information Management for z/OS Messages and Codes, GC31-8748-00, contains the messages and completion codes issued by the various Tivoli Information Management for z/OS applications. Each entry includes an explanation of the message or code and recommends actions for users and system programmers.

Tivoli Information Management for z/OS Operation and Maintenance Reference, SC31-8749-00, describes and illustrates the BLX-SP commands for use by the operator. It describes the utilities for defining and maintaining data sets required for using the Tivoli Information Management for z/OS licensed program, Version 7.1.

Tivoli Information Management for z/OS Panel Modification Facility Guide, SC31-8750-00, gives detailed instructions for creating and modifying Tivoli Information Management for z/OS panels. It provides detailed checklists for the common panel modification tasks, and it provides reference information useful to those who design and modify panels.

Tivoli Information Management for z/OS Planning and Installation Guide and Reference, GC31-8751-00, describes the tasks required for installing Tivoli Information Management for z/OS. This book provides an overview of the functions and optional features of Tivoli Information Management for z/OS to help you plan for installation. It also describes the tasks necessary to install, migrate, tailor, and start Tivoli Information Management for z/OS.

Tivoli Information Management for z/OS Problem, Change, and Configuration Management, SC31-8752-00, helps you learn how to use Problem, Change, and Configuration Management through a series of training exercises. After you finish the exercises in this book, you should be ready to use other books in the library that apply more directly to the programs you use and the tasks you perform every day.

Tivoli Information Management for z/OS Program Administration Guide and Reference, SC31-8753-00, provides detailed information about Tivoli Information Management for z/OS program administration tasks, such as defining user profiles and privilege classes and enabling the GUI user interface.

Tivoli Information Management for z/OS Reference Summary, SC31-8754-00, is a reference booklet containing Tivoli Information Management for z/OS commands, a list of p-words and s-words, summary information for PMF, and other information you need when you use Tivoli Information Management for z/OS.

Tivoli Information Management for z/OS Terminal Simulator Guide and Reference, SC31-8755-00, explains how to use terminal simulator panels (TSPs) and EXECs (TSXs) that let you simulate an entire interactive session with a Tivoli Information Management for z/OS program. This book gives instructions for designing, building, and testing TSPs and TSXs, followed by information on the different ways you can use TSPs and TSXs.

Tivoli Information Management for z/OS User's Guide, SC31-8756-00, provides a general introduction to Tivoli Information Management for z/OS and databases. This book has a series of step-by-step exercises to show beginning users how to copy, update, print, create, and delete records, and how to search a database. It also contains Tivoli Information Management for z/OS command syntax and descriptions and other reference information.

Tivoli Information Management for z/OS World Wide Web Interface Guide, SC31-8757-00, explains how to install and operate the features available with Tivoli Information Management for z/OS that enable you to access a Tivoli Information Management for z/OS database using a Web browser as a client.

Other related publications include the following:

Tivoli Decision Support: Using the Information Management Guide is an online book (in portable document format) that can be viewed with the Adobe Acrobat Reader. This book is provided with Tivoli Decision Support for Information Management (5697-IMG), which is a product that enables you to use Tivoli Information Management for z/OS data with Tivoli Decision Support. This book describes the views and reports provided with the Information Management Guide.

IBM Redbooks™ published by IBM's International Technical Support Organization are also available. For a list of redbooks related to Tivoli Information Management for z/OS and access to online redbooks, visit Web site **http://www.redbooks.ibm.com** or **http://www.support.tivoli.com**

Index

P

Q

R

S

T

U

W

File Number: S370/30xx/4300
Program Number: 5697-SD9

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8757-00