BusinessLink Technical Services

Strategi
HSM Programmer's Guide

Version v1r7

This manual applies to Strategi version v1r7m0 and later, and was last revised in June, 2000.

ADVANCED BusinessLink Corp. may have patents and/or patent pending applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

Every effort has been made to ensure the accuracy of this manual. However, ADVANCED BusinessLink Corp. and Advanced BusinessLink (Australia) Pty. Ltd make no warranties with respect to this documentation, and shall not be liable for any errors, or for incidental or consequential damages in connection with the performance or use of this manual, or the examples pertaining to products and procedures as described herein. The information in this manual is subject to change without notice.

Other trademarks, trade names and brand and product names used in this manual are trademarks or registered trademarks of their respective holders.

Printed in the United States of America.

A Letter from the CEO

Thank you for your interest in ADVANCED BusinessLink Corp. and our Strategi™ software.

Strategi is the technology-leading, 100% native AS/400 e-business solution. It is a suite comprising four separate "best-of-breed" components, each architected around BusinessLink's philosophy of providing firms with the technological edge that will separate them from their competitors.

While the components are unique and exciting in themselves, Strategi is much more. It was designed and architected to address the business issues faced in the development, deployment, and management of an e-business initiative. By combining all the Strategi components, BusinessLink provides a single environment that enables you to build a bridge to the past (legacy apps through the Java applet) and a bridge to the future (HSM eBusiness applications).

This is why we believe you will call it your ebusiness Strategi.

Sincerely,

Chris Lategan
Chief Executive Officer
ADVANCED BusinessLink Corp.

A Note to Readers of this Manual

Built on the premise that technological solutions are useless unless they provide real-world business benefits, Strategi has been architected to provide your organization a foundation to enable creative breakthrough e-business solutions. This manual has been designed to enhance your usability experience with Strategi as well.

The latest versions of this document and other Technical Support Bulletins can be downloaded from ADVANCED BusinessLink Corp.'s Support Website, http://support.businesslink.com.

You may print this in duplex format using Adobe's Acrobat Reader, which is available for download from http://www.adobe.com/products/acrobat/readstep.html. The latest version as of this writing is v4.05; earlier versions of Acrobat Reader may not support duplex printing.

With some installs of Adobe Acrobat, your printer may not resolve the characters correctly, and once printed, all characters will appear as a rectangles or as symbols. If this happens, you will need to select "Print as image" from the Acrobat print dialogue. This will cause the print to occur correctly.

If you have any questions, comments or suggestions, please feel free to contact either myself at the email address below, or the various local branches of the BusinessLink Technical Support Division at the phone numbers and email addresses in Appendix 1.

Sincerely,

BusinessLink Technical Services

Contents

A Letter from the CEO 3

A Note to Readers of this Manual 4

HSM Specification Information 8

INCOMPATIBILITIES 8
CHANGES TO HSM PROGRAMING SPECIFICATION 8

Overview 11

Strategi High Speed Messaging 11
HSM's Relevance to Your Business 11
Client/Server Interaction 12
HSM System Requirements 12
AS/400 Requirements 12
PC Requirements 12

A Practical Example 13

Combining Theory with a Working HSM Application 13
Client/Server Interaction 13
Server Logic 15

Server Design 16

Design Overview 16
Receiving & Processing Client Requests 17
Design Objectives 18
Server Message Elements & Interface Calls 18
Server Behavior on Error 20

HTML Implementation 21

Overview 21
Visitor Counter Example 21
Guest Book Example 23
Stock Inquiry Example 23

HTML Reference 25

HTML Keywords 25
<HSM NAME=field_name> </HSM> 25
<HSMBLOCK NAME=block_name> </HSMBLOCK> 26

SHTML & Server Side Includes 27
.SHTML and .SHTM File Extensions 27
Server Side Includes 28

HSM Resource File Reference 29

File Construct 29
Groups 29
[SERVER REQUEST] 29
[REPLY] 29
[DO] 29
[BLOCK] 29
[SUBMIT BUTTON] 30
Keywords and Values by Group 30
[SERVER REQUEST] 30
[REPLY] 30
[DO] 31
[BLOCK] 31
[SUBMIT BUTTON] 31
Additional Keywords and Considerations 32
Conditions 32
Condition Criteria 32
Explicitly Setting Field Values 33
Exporting Embedded Values 33
Array Elements 34
Naming Conventions 34
Automatic HSM Start Position 34
Special Values 35
INDEX Special Values 36
HTTP Header Special Values 37
Conditional [DO] 37
Switching to Another File 37
Values in Link URLs 38
Returning HTTP Messages 39
Error Messages 39
HTTP Error redirection through a URL 39

HSM File Upload & Pushfeed 40

Strategi Pushfeed 40
Push OPcodes (For Strategi Pushfeed) 41
LSTPUSH 41
DLTPUSH 42
HSM File Upload 43
File Property OPcodes (For HSM File Upload) 45
SETPROH 45
CHGPROH 45
GETPROH 46
FILEINFO 46
Distributed HSM Servers (DHSM) 48

Other Topics 51

Other Resources 51
SGIEXAMPLE 51

http://support.businesslink.com 51
Additional Questions 51

Appendix 1, Converting Old Servers 53

Principle Benefits 53
COMPATABILITY INTERFACE Conversion 54
ILE INTERFACE Conversion 56

Appendix 2, Additional HSM Administrative Features 59

HSM Authorities 59
HSM Performance Monitoring 59
Customized Login Pages 60
HSM Server Packaging & Installation Commands 61

HSM Specification Information

INCOMPATIBILITIES

There are no known incompatibilities in Strategi V1R6M0 that will affect the operation of existing servers. There are new features existing servers can use, if they enable the feature.

There is however, a subtle change in the way authorization is done for server call chains. This is the situation where a client makes a request of a server, and this server in turn makes a request of another server. This change should not present a problem to any existing systems.

Servers are always passed the original details (client-type, client-id, client location and user-attribute). But authority is always checked against the immediate client: server-a checks client, server-b checks server-a.

CHANGES TO HSM PROGRAMING SPECIFICATION

Changed Function: HSMRCVRQS & hsm_rcvrqs()
This function has the altered spec below. There are additional optional fields that provide extra information on the request.

```
HSM_BOOL hsm_rcvrqs(
HSM_BYTE       *opcfld,     /* HSM_SIZOPC - Operation Code          */
void           *dtafld,     /* HSM_SIZDTA - Operation Data          */
HSM_DEC4       *dtalenp,    /*            - Operation Data Length   */
HSM_BYTE       *clttyp,     /* HSM_SIZTYP - Client Type             */
HSM_BYTE       *cltfld,     /* HSM_SIZCLT - Client ID               */
...);
Optional Parms:
HSM_BYTE       *locfld      /* HSM_SIZLOC - Client Location         */
HSM_BYTE       *usratr      /* HSM_SIZUSRA - User Attribute         */
```

Calls to hsm_svropt() are used to enable the optional fields - the fields are expected in the same order as the calls to hsm_svropt(). Hence if a server was interested only in the user attribute, a single call to hsm_svropt() would be required:

```
hsm_svropt("SETRQSPARM", "USRATR");
hsm_rcvrqs(OPcode,data,length,client_type,client_id,user_attr);
```

But if the location and user attribute were desired:

```
hsm_svropt("SETRQSPARM", "CLTLOC")
hsm_svropt("SETRQSPARM", "USRATR");
hsm_rcvrqs(OPcode,data,length,client_type,client_id,client_location,user_attr);
```

And the order is defined by the order of calls to hsm_svropt(), so:

```
hsm_svropt("SETRQSPARM", "USRATR");
hsm_svropt("SETRQSPARM", "CLTLOC");
hsm_rcvrqs(OPcode,data,length,client_type,client_id,user_attr,client_location);
```

And, lastly, if the defaults are used:

```
hsm_rcvrqs(OPcode,data,length,client_type,client_id);
```

In RPG the above calls to hsm_svropt() look like:

```
CALLB       'HSMSVROPT'
PARM        'SETRQSPARM'          TMPCHR20      20
PARM        'CLTLOC'              TMPCHR500     500
```

New Function: HSMSVROPT & hsm_svropt()
These new functions (for RPGLE & CLE respectively) generically enable various server-side options. The interpretation of optval is dependent on optkwd. Keywords and values are case-insensitive. This function is available with an ILE interface only. As with other

```
HSMSVROPT
HSM_BYTE      OPTKWD[20]
HSM BYTE      OPTVAL[500]

hsm_svropt(
HSM_BYTE      *optkwd,
HSM_BYTE      *optval);
```

KEYWORD	VALUE
SETRCVTMO	Number of seconds
SETRQSPARM	Parameter Name
LICFEATURE	Feature Code

SETRCVTMO: Enables the hsm_rcvrqs() function to periodically return with OPCODE *TIMEOUT. This OPcode should NOT be replied to. The server can make periodic checks before returning to the hsm_rcvrqs() all. This value is -1 by default, indicating no timeout.

SETRQSPARM: Enables optional request fields. To define a different set of optional fields use one or more calls to set this server option. The value is a literal field name: "CLTLOC" or "USRATR". Only one set of calls is permitted for SETRQSPARM. Attempting to do a SETRQSPARM after having done an hsm_rcvrqs() will generate an error (terminating the server).

New Feature: User Attribute
The server configuration can now define a user attribute. When a Strategi user accesses the server the new user-attribute field is populated with the contents of the specified user attribute for that user (blank if no attribute defined). In this way our customers can automatically and efficiently associate users with an external reference for their server. The user attribute is picked up on the first request in a chain; all subsequent requests pass this original user attribute along. The request chain may include DHSM requests to a peer system. (i.e. For: [client server-a -> server-b -> server-c] The user-attribute is only picked up for the request to server-a; server-b and server-c both get this original attribute).

New Feature: Client Location
With the introduction of DHSM peer networking, it is possible that the request did not originate from this system. The new optional field CLTLOC allows the server to know the peer system code that the request originated from. When the request originated from the same system, *LOCAL is passed in this field.

Note that this system code is taken from the first remote system, so if [client-> server-a -> server-b] and if client, server-a and server-b are all on different systems, the client location is the system code defined on the system for server-a. In such a setup (which would be obscure and probably unnecessary as client could probably talk directly to server-b) the peer tables should use the same system code on all systems.

Overview

Strategi High Speed Messaging

Strategi High Speed Messaging (HSM) is a groundbreaking new communications technology. As the foundation for high-speed transaction processing, HSM frees you from the pain of developing and testing CGI-BIN alternatives and provides a solid, integrated business solution.

Packaged as high performance “middleware”, High Speed Messaging blends seamlessly with Strategi to connect remote clients to your AS/400 over the Internet.

HSM's Relevance to Your Business

High Speed Messaging has been designed to enable your business to move from yesterday's legacy applications into today's (and tomorrow’s) interactive applications. Your customers and staff need no longer put up with the awkward key-mappings forced on them by a 5250 interface, the unnatural GUI behavior necessitated by a screen scraper, or the slow response times with unreliable connectivity inherent in “do-it-yourself” communications programming.

Instead of “Green Screens“, “Screen Scrapers“ or detailed communications programming, Strategi’s High Speed Messaging makes it possible to produce striking graphical applications which communicate effortlessly with your AS/400.

HSM utilizes the AS/400 as a powerful back-end server while providing total control over the client interface, giving you the freedom to create stunning, high-performance applications that “stand out in the crowd” and leave your customers and your competition wondering how you achieved them.

Strategi’s High Speed Messaging allows authorized users to converse with your AS/400 via the Internet from any Web browser, without actually letting the user onto your AS/400.

How? HSM acts as a “middleman” between the Internet and your AS/400, limiting the user’s interaction with the 400 to the tasks you have programmed into your HSM server. This means that at no time can the end user have any interaction with your system that the HSM server has not already been designed to allow. And, using Strategi’s HTTP authentication, you can further limit the user’s interaction by accessing their registration number for authority verification.

Client/Server Interaction

Strategi includes a unique delivery transport for High Speed Messaging. Via HTML, Strategi provides a client/server architecture for seamless and communications-layer-transparent messaging with an AS/400 program. A web page on your Strategi-hosted web site makes a call to the Strategi subsystem on the AS/400. Almost instantly, the message is relayed by Strategi to your HSM application on the AS/400. Then, your HSM application processes your request and calls the reply API. Strategi then handles relaying the AS/400 program's reply back to the next web page.

HSM System Requirements

AS/400 Requirements

- OS/400 V4R2M0 or later running Strategi.
- LAN connections as required by Strategi.
- Approximately 45 MB of AS/400 disk space to run Strategi.

PC Requirements

- Web browser

A Practical Example

Combining Theory with a Working HSM Application

The following example is a basic HSM implementation, yet displays striking results.

The software “Client” is written with HTML and HSM Resource files and runs in a web browser. It communicates with the AS/400 by calling functions in Strategi’s communications routines. The “client” has no specific knowledge of the connection, the server logic or the AS/400; that's all handled in the background by Strategi.

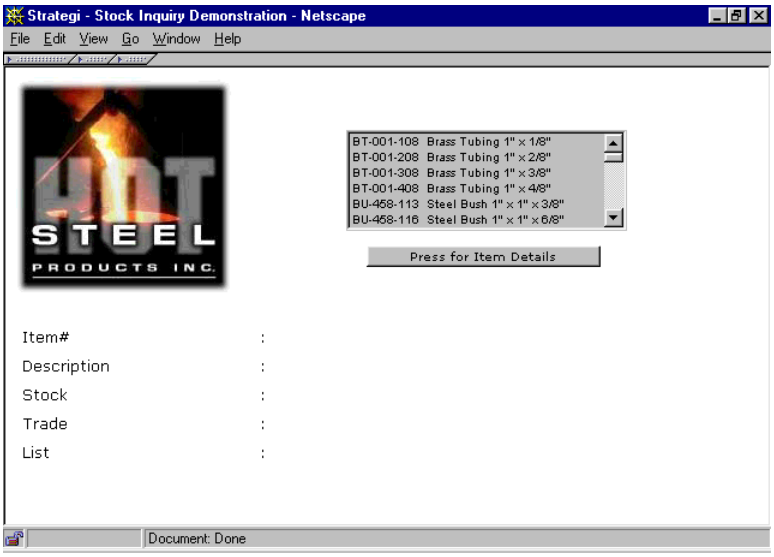
The “Server” is can be written in RPG, CL, or any other programming language on an AS/400. It is loaded, run and managed by Strategi, and has no knowledge of how it connects to clients or even how client requests get sent to the HSMRequest() API. Further, it has no program logic or knowledge of how clients get the replies it sends with the HSMReply() API, or how the correct reply gets back to the correct client. It doesn't need to; Strategi takes care of everything.

Client/Server Interaction

The following demonstrates a typical interaction of HTML client and server. Some detail has been omitted to simplify the explanation, since the objective here is to provide a feel for HSM interaction rather than detailed program logic or client HTML programming.

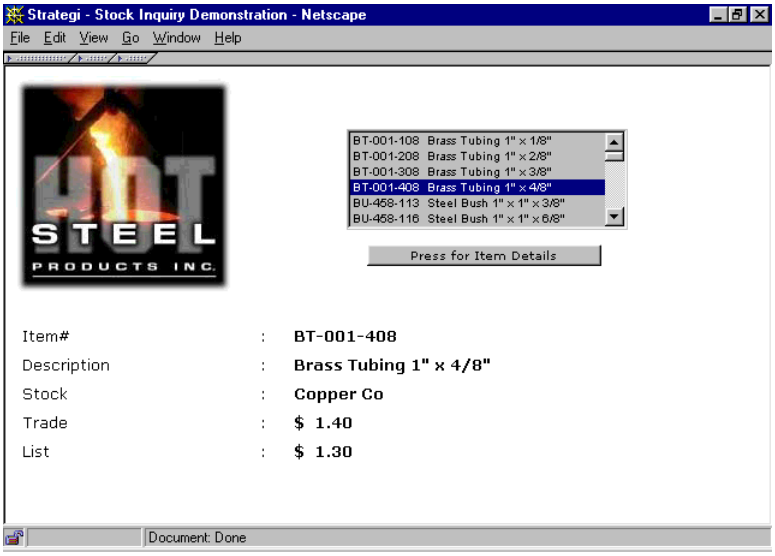
Please be aware that when we speak of a “client” in Strategi, it is somewhat of a misnomer, as there is no software-installed-on-the-PC “client” per se, other than the browser. We refer to the HTML web page served up by the web browser as the “client”, in order to keep the “client/server” terminology in order.

After going to the “client’s” webpage, the main form (pictured below) is displayed, then:



The user selects an item in the list and presses the "Details" button - this results in a call to HSMRequest(), with an OPcode of GETITEM. The item number selected is passed as a field in the DataBlock. The server returns a reply DataBlock via HSMReply(), which is then parsed into fields by the HSM Resource File. The fields are then filled in the HTML template, to be displayed in the list box (see example below). This step could be repeated several times as the user examines pricing and availability for different items.

If you look at the source HTML for any webpage returned with AS/400 data in it, and compare it with the raw /homepage.htm, you can see how the AS/400, through the Strategi HSM server, seamlessly inserts the appropriate data from the HSMReply() into the HTML.



In this example, all data, including the list box of items and the pricing values, are coming live from the AS/400. No data is stored locally on the PC.

The end-user has no idea the data is actually on the AS/400, regardless of how far away the AS/400 actually is, since performance is similar to other PC-based applications (where data is actually stored locally on the PC hard disk).

And of course, logic for pricing is retained on the AS/400, rather than being re-coded in an unfamiliar PC language.

Server Logic

The server logic can be pseudo-coded as:

```
Do
Receive Request
if OPcode = "GETLIST"
       Read List of Items from Master File
       Put item list into reply structure
       End If
If OPcode = "GETITEM"
       Validate Item number passed from Client
       Read Item Warehouse record to get stock, $, etc.
       Put values into reply structure
       End If
[.. test for & process *STOP & *PING ..]
Send Reply
Loop
```

The server logic is abstracted, so it can simultaneously interact with several clients. It is basic "transaction processing", and retains no client specific data between requests.

Server Design

Design Overview

Server design is fairly easy to sketch out. A minimum server is about a dozen lines of CL code, and is a *PGM object expecting one 10-character parameter, the server name.

The Server program then receives, processes and replies to requests received from client(s).

Prior to being used, the HSM Server definition must be created, at which time its creator designates a Server Name by which the HSM client will request its services (please refer to the Strategi Administration Menu, “Work with HSM Servers“, for examples).

The server is essentially a transaction processor. Its specification could broadly be pseudo-coded as:

```
Main Line
Clear STOP-FLAG
While STOP-FLAG is not set {
Clear Request & Reply Elements
Receive Request
       Test OPcode {
              OPcode = "*PING" {
                     Place server description in reply data structure
                     Return *PING OPcode
                     }
              OPcode = "*STOP” {
                     if (Client Type is "*CONTROL") {
                            Move "*STOPPED" to reply OPcode
                            Set STOP-FLAG.
                            }
                     }
              OPcode = “(Application Specific OPcode)” {
                     Application processing
                     }
              If no OPcodes Recognized {
                     Move "*ERROR" to reply OPcode
                     Move error detail to reply data
                     }
              }
       Send Reply
       }
Exit PGM
```

Receiving & Processing Client Requests

Certain OPcodes have a system-defined meaning:

***STOP**
Indicates server shutdown is being requested by Strategi, either in response to a system administrator request or to a general system shutdown.

Servers should only respond to stop it the client code of the request is "*CONROL", which will only be set when the server is shut down properly. Servers must respond to *STOP by setting the reply OPcode of "*STOPPED", and also setting an indicator such that, after the reply was sent, the loop would break and the program can exit.

***PING**
Servers must reply to *PING requests with a corresponding OPcode of *PING, and a DataBlock containing brief text describing the server's function, purpose or summary of services, and a version number (which allows the client to "ping" the server and check that is connecting to a server it is compatible with).

Strategi does not dictate the length of this text. Since it is used on many list-type displays, and may be used by the client's programs to display to end users, it is recommended the text be limited to 40 characters, to facilitate practical screen design. The ping reply text can be used as a descriptive title to the end user, letting them see exactly what server and version of that server they are connecting to.

***ERROR**
In the event that the server is passed an OPcode which it does not check for, its catch-all reply should be *ERROR. *ERROR should reply in a very specific manner. The first seven bytes of the reply data should contain the server name, followed by one space, and then up to 200 bytes of description about the error, for instance "The OPcode was not recognized".

***NOSERVER**
This OPcode will be returned by Strategi if the server requested is not currently running. This should not be included in your server program code, since this should never be returned if the server is actually running.

All other OPcodes are user defined, predetermined at the time of Server design and used by client application. The only restrictions on naming are that the OPcode can be no longer than ten characters, and cannot start with "*". "*" Is reserved for Strategi's required OPcodes, present and future.

A strict request/reply protocol must be observed between Client and Server. This means for every Client request there is a resulting reply from the Server.

Data transferred between the Client and Server is automatically translated between EBCDIC and ASCII by Strategi. Because of this, data should be restricted to Alpha/Numeric and basic symbols (!, @, #, ...)

Design Objectives

You should keep the following objectives in mind when considering HSM Server design:

1. "Black Box" the Strategi system (and therefore the AS/400) from the outside world. Never require clients to have specific knowledge of underlying file structures, database formats, etc.

   For example, instead of requiring clients to ask for data from specific files (such as getting "10 item master records"), abstract these requests to just ask for the objects ("get 10 items").

2. Design the server to be "stateless", i.e., not retaining specific knowledge from request to request. This will allow it to be safely "Parallel-Tasked".

   For example, if the client is creating an order which may be comprised of several transactions then requires final confirmation of the order, have the server return a "handle" to the client following a "Create Order" request.

   This may actually be an order header key but to the client, it is just an abstract handle, which it is required to pass to the server on subsequent requests ("Add Line Item", etc.). The server can then safely deal with multiple clients, and need not be concerned with continuity of conversation, since it will not have any client-specific variables stored beyond the life of an individual transaction.

3. Rationalize client requirements to a set of structured requests, which the server (or servers) can reply to individually.

   For example, the "Hot Steel" demonstration earlier in this manual structured an entire product inquiry to just two structured requests: GETLIST and GETITEM.

Server Message Elements & Interface Calls

Message Elements

Receiving a Request

```
          OPcode          10     The request operation code
          Data            9999   The request data
          Data Length     4.0    The request data length
          Client Type     10     The type of client making the request
          Client          10     The client making the request
```

Sending a Reply

```
          OPcode          10     The reply operation code
          Data            9999   The reply data
          Data Length     4.0    The reply length
```

Server Interface Calls

There are two bound interface sets, and two external interface sets.

Bound Set 1, for C programs

```
hsm_rcvrqs(
       HSM_BYTE       *OPcode,       /*   10 */
       void           *data,         /* 9999 */
       decimal(4)     *data_len
       HSM_BYTE       *client_type,  /*   10 */
       HSM_BYTE       *client,       /*   10 */
       );

hsm_sndrpy(
       HSM_BYTE       *OPcode,       /*   10 */
       void           *data,         /* 9999 */
       decimal(4)     data_len
       );
```

Bound Set 2, for ILE RPG/CLP programs

```
HSMRCVRQS
       OPCODE         *CHAR   10
       DATA           *CHAR 9999
       DATA_LEN       *DEC     4
       CLIENT_TYPE    *CHAR   10
       CLIENT         *CHAR   10

HSMSNDRPY
       OPCODE         *CHAR   10
       DATA           *CHAR 9999
       DATA_LEN       *DEC     4
```

External Set 1, for OPM Programs

```
HSMRCVRQ
       OPCODE         *CHAR   10
       DATA           *CHAR 9999
       DATA_LEN       *DEC     4
       CLIENT_TYPE    *CHAR   10
       CLIENT         *CHAR   10

HSMSNDRP
       OPCODE         *CHAR   10
       DATA           *CHAR 9999
       DATA_LEN       *DEC     4
```

External Set 2, for old-specification OPM server programs

```
XRCVDTAQ
       DTAQ           *CHAR   10 /* THIS PARM IGNORED BY HSM */
       LIB            *CHAR   10 /* THIS PARM IGNORED BY HSM */
       RCDLEN         *DEC     5 /* THIS PARM IGNORED BY HSM */
       RCD            *CHAR 9999
       TIMEOUT        *DEC     5 /* THIS PARM IGNORED BY HSM */

XSNDDTAQ
       DTAQ           *CHAR   10 /* THIS PARM IGNORED BY HSM */
       LIB            *CHAR   10 /* THIS PARM IGNORED BY HSM */
       RCDLEN         *DEC     5 /* THIS PARM IGNORED BY HSM */
       RCD            *CHAR 9999

DATA QUEUE MESSAGE STRUCTURE
       RESERVED1      *CHAR   11
       USER           *CHAR    9 /* first 9 char of user, only */
       RESERVED2      *CHAR   12
       SERVER         *CHAR   10
       OPCODE         *CHAR    8 /* first 8 char of OPcode only */
       DATA_LEN       *CHAR    4
       DATA           *CHAR 9945 /* first 9945 char of data only */
```

NOTE 1: Calling the bound functions is faster than calling the external programs.

NOTE 2: The external API's are *programs* not commands.

NOTE 3: Maximum record length in BusinessLink CoProcessor-based HSM was 1920, maximum record length in Strategi prior to v1r3m1 was 4150.

Server Behavior on Error

When the server encounters an error condition it will be terminated. The possible conditions which can cause this are:

- Any failure to create the message queue object.
- Receive request or Send reply with an invalid argument (NULL pointer).
- Receive request when no reply to previous OPcode is done.
- Any reply before receiving a request.
- Any reply with an invalid OPcode (blank or non-std beginning with '*').

This means that the server has no error/exception conditions to deal with. When the server returns from acting upon RCVRQS it will have an OPcode to process. It must reply to that OPcode before attempting another RCVRQS.

HTML Implementation

Overview

To simplify the interface for programming Strategi HSM in HTML, Strategi uses a template-based approach, where HTML pages are templates and the dynamic generation of the final HTML page is done "on the fly" by the Strategi server.

The web server, using specifications in a special file (called a HSM Resource File), obtains the fields to be substituted in the template. This file is a text file in industry standard .INI format, and is named "<filename>.HSM", where <filename> is the same as the name of the .HTM template intended to be processed. It must exist in the same directory as the template .HTM document, and its presence indicates to the Strategi server that the .HTM (of the same name) is not a static web page, but requires dynamic substitution. This has a two-fold objective:

- Define HSM resources required by the HTML page such as structuring input fields, making HSM server calls and defining output fields which can be mapped into template HTML documents.
- Keeping the HTML document as standard as possible, free from detailed and cryptic substitution instructions, etc.

Visitor Counter Example

The first example is fairly basic, showing how the one might create a text based counter on the home page:

HTML (homepage.htm)

```
<HTML>
<TITLE>Hot Steel Products Inc. Home Page</TITLE>
<H1>Hot Steel Products Inc.</H1>
Welcome to our Web Site<BR>
You are visitor number <HSM NAME=VISITOR_NUMBER>
</HTML>
```

HSM (homepage.hsm)

```
[SERVER REQUEST]
SERVER=WEBTOOLS
OPcode=COUNTER

[REPLY]
OPcode=COUNTER
Start_Length_Field=1,9,VISITOR_NUMBER
```

The HTML template is standard HTML with the exception of the special HSM tag.

This tag names a field (VISITOR_NUMBER) which the HSM Resource File defines as being supplied in a 9-byte reply from server WEBTOOLS, OPcode COUNTER.

This, in turn, decides which server is called, selects the OPcode and defines any data required to fully qualify the request. Strategi makes the server request and maps the reply data to the VISITOR_NUMBER field as specified.

Many of the things "defaulted" in the example above are better illustrated through the following example:

HTML (homepage.htm)

```
<HTML>
<TITLE>Hot Steel Products Inc. Home Page</TITLE>
<H1>Hot Steel Products Inc.</H1>
Welcome to our Web Site<BR>
You are visitor number <HSM NAME=VISITOR_NUMBER>
</HTML>
```

HSM (homepage.hsm)

```
* Call server to get next visitor number

[SERVER REQUEST]
SERVER=WEBTOOLS
OPcode=COUNTER

* Evaluate replies

[REPLY]
OPcode=COUNTER
Start_Length_Field=1,9,VISITOR_NUMBER

[REPLY]
OPcode=*NOSERVER
Substitute_URL=noserver.htm

[REPLY]
OPcode=*OTHER
Assign_Field_Value=unexpectedOPcode,*REPLY.OPCODE
Start_Length_Field=1,*,unexpectedData
Substitute_URL=unexpect.htm
```

Note that both HSM Resource file examples will have the same effect, provided the server is active and responds as expected with an OPcode of COUNTER. The second example caters to the situation where the server is not running, and the situation where the server returns an OPcode other than COUNTER for some reason.

The HTML file requested by the browser will be returned immediately following processing of the resource file, unless a Substitute_URL= statement is invoked, in which case the specified URL will be returned instead.

Guest Book Example

Following is a simple guest book application, where users can enter their name and email address, then press a button to "sign" the guest book. All data is written to the AS/400 database by the WEBTOOLS server:

HTML (guestbk.htm)

```
<HTML>
<TITLE>Hot Steel Products Inc.</TITLE>
<H1>Guest Book</H1>
<FORM METHOD=POST ACTION="addentry.hsm">
Your Name:<BR>
<INPUT NAME="GUEST_NAME" TYPE="TEXT" SIZE="40"><BR>
Email Address:<BR>
<INPUT NAME="EMAIL_ADDRESS" TYPE="TEXT" SIZE="40"><BR>
<INPUT NAME="SIGN_IT" TYPE="SUBMIT" VALUE="Sign It">
</FORM>
</HTML>
```

HSM (addentry.hsm)

```
* Call server add visitor details to guest book

[SERVER REQUEST]
SERVER=WEBTOOLS
OPcode=ADDGUEST
Start_Length_Field=1,40,GUEST_NAME
Start_Length_Field=41,40,EMAIL_ADDRESS

* Evaluate replies

[REPLY]
OPcode=ADDED
Substitute_URL=added_ok.htm
```

Please note the filename of the .HSM Resource file is ADDENTRY.HSM, not GUESTBK.HSM (as in the previous example). This is correct, since GUESTBK.HTM is not a template, and therefore does not require any HSM processing to be displayed.

HSM resources are only required to process the form, so ADDENTRY.HSM is specified as the target of the submit button. There does not need to be an actual ADDENTRY.HTM, since only HSM resources (and not the display of an HTML document) are required. The document to be returned to the browser is specified as Substitute_URL. If one was not otherwise specified, or no reply OPcode matched one specified, an error would be displayed.

Stock Inquiry Example

In this example, there is one main .HTM and two .HSM resource files. The first HSM resource file has the same name as the .HTM, and would therefore be processed prior to displaying the .HTM page. The second HSM resource is called if the form is "submitted", i.e. if a submit button for the form is pressed.

HTML (stock.htm)

```
<HTML>
<H1>Stock Inquiry</H1>
<FORM METHOD=POST ACTION="stock_fm.hsm">
<SELECT NAME="SELECTED_ITEM" SIZE=10>
<HSMBLOCK NAME=ITEMLIST_BLOCK>
  <OPTION><HSM NAME=ITEMS>Pipes
    <OPTION>Rods
    <OPTION>Plates
    <OPTION>Rods
    <OPTION>Plates
    <OPTION>Rods
    <OPTION>Plates
  </HSM>
</HSMBLOCK>
</SELECT>
<INPUT NAME="Prev10" TYPE="SUBMIT" VALUE="Previous 10">
<INPUT NAME="Next10" TYPE="SUBMIT" VALUE="Next 10">
<BR><BR>
<INPUT NAME="GetDetails" TYPE="SUBMIT" VALUE="Item Details">
<BR><BR>
<HSMBLOCK NAME=DETAILS_BLOCK>
Stock available is <HSM NAME=ONHAND>44</HSM><BR>
Stock on Backorder is <HSM NAME=BACKORDERED>44</HSM><BR><BR>
</HSMBLOCK>
<INPUT NAME="order" TYPE="SUBMIT" VALUE="Place an Order">
</FORM>
```

HSM (stock.hsm)

```
* Call server to get preload info for stock

[SERVER REQUEST]
SERVER=STOCK
OPcode=GETDATA
Start_Length_Field=1,1,OP
Start_Length_Field=2,10,POSITIONING_KEY
Export_Field=LIST_COUNT
Export_Field=ITEMS

* Evaluate replies

[REPLY]
OPcode=ITEMLIST
Start_Length_Field=1,4,LIST_COUNT
Start_Length_Field=5,10x20,ITEMS

[REPLY]
OPcode=ITEMDATA
Start_Length_Field=1,20,DETAIL_ITEM
Start_Length_Field=21,5.0,ONHAND
Start_Length_Field=26,5.0,BACKORDERED

* Block Handling

[BLOCK]
Name=ITEMLIST_BLOCK
Condition_Variable=LIST_COUNT
```

```
Condition_Criteria=NONZERO
Repeat_Count=LIST_COUNT

[BLOCK]
Name=DETAILS_BLOCK
Condition_Variable= DETAIL_ITEM
Condition_Criteria=NONBLANK
```

HSM (stock_fm.hsm)

```
* Process Stock form

[SUBMIT BUTTON]
Name=Prev10
Assign_Field_Value=OP,"B"
Assign_Field_Value=POSITIONING_KEY,ITEMS[1]
Substitute_URL=stock.htm

[SUBMIT BUTTON]
Name=Next10
Assign_Field_Value=OP,"F"
Assign_Field_Value=POSITIONING_KEY,ITEMS[10]
Substitute_URL=stock.htm

[SUBMIT BUTTON]
Name=GetDetails
Assign_Field_Value=OP,"I"
Assign_Field_Value=POSITIONING_KEY,SELECTED_ITEM
Substitute_URL=stock.htm

[SUBMIT BUTTON]
Name=Order
Export_Field=SELECTED_ITEM
Substitute_URL=order.htm
```

HTML Reference

HTML Keywords

<HSM NAME=field_name> </HSM>

This <HSM> tag defines a field in the HTML template, which is substituted by the field's actual value prior to the page being served to the requesting browser. An HSM field name can be up to 31 characters long.

The closing </HSM> tag is optional, and permits sample text to be inserted at template design time. Any data between the <HSM NAME=…> tag and the following </HSM> will not be included in the final generated HTML. Sample text unrelated to a field substitution can be marked with a "<HSM> …sample… </HSM> pair", omitting the NAME= keyword.

Following a <HSM> tag, the closing </HSM> is deemed to be omitted if another tag beginning <HSM> is found prior to finding a </HSM>, since they cannot be nested (but can enclose other HTML tags).

There is an additional attribute of the <HSM> tag, ESCAPE, eg "<HSM NAME=VAR ESCAPE=HTML>". This can be used to accomplish different sorts of escaping for the data. This attribute is not required. The four possible values of ESCAPE are:

HTML	Normal HTML escaping, eg "&lt"; for "<". This is the default.
URL	URL escaping, eg "%3C" for "<".
BACKSLASHX	Used for substitution into Javascript literals, eg "\x3C"for "<" all but alphanumeric escaped.
BSX	Same as BACKSLASHX.
BACKSLASHX25	Used for URL arguments of Javascript functions that are part of an anchor HREF (where simple URL escaping is not enough because some browsers wrongly do a URL-style unescape before passing the value to the function, and the function itself cannot do the escaping if URL data is involved).
BSX25	Same as BACKSLASHX25.
NO	No escaping will be done – data is already valid for context.
NONE	Same as NO.

<HSMBLOCK NAME=block_name> </HSMBLOCK>

The <HSMBLOCK> tag and its following </HSMBLOCK> perform no substitution of their own, but define the HTML statements they enclose as a "block", which can be manipulated by name in the HSM resource file. They can be used to optionally exclude blocks of conditional code, and to repeat lines of generated code, such as items in a list box, etc.

The closing </HSMBLOCK> tag is mandatory.

Conditional <HSMBLOCK> tags can be nested but cannot overlap. Repeat <HSMBLOCK> tags cannot be nested. An example would be a conditional block containing a repeat block, containing further conditional blocks, with the outer conditional preventing table headers for an empty table, and the inner conditionals preventing empty table rows.

If the block name has not been defined in the associated .hsm file(s), the tag and its matching end tag are ignored, so the HTML it encloses will be displayed only once and exactly as specified in the template HTML.

Note that all variables within a repeat block are automatically indexed, reverting to the simple variable name if the indexed element is undefined.

There are limits to the size of blocks and sample text, generating error messages if exceeded. An HSM repeat block, including both start and end tags, is limited to 7000 characters of original HTML. Other HSM blocks, and HSM sample text, are limited to only 1000 characters. Exceeding the first will cause a "Repeat block too large" error; others may simply act as if the ending tag does not exist.

There is no limit to the amount of HTML generated and sent to the browser, only in the HTML before substitutions occur. The HSMBLOCK name cannot exceed 31 characters in length.

SHTML & Server Side Includes

.SHTML and .SHTM File Extensions

Sometimes simple HSM substitution does not require use of an HSM file. For example, you may have passed a value to the HTML file through a URL (see "Values in Link URLs"), and simply need to display that value on the HTML page.

In this case, you would use the extension "shtml" or "shtm" instead of "html" or "htm". SHTML files are processes by HSM in the same manner as HTML files. They will also be displayed just as HTML files by your browser.

Server Side Includes

Server Side Includes are a very useful way to save time when writing HTML. A Server Side Include is one tag added to an HTML file that points to another HTML file. The contents of this second HTML file are added to the first HTML file in place of the #Include tag.

For example:

HSM (homepage.hsm)

```
[DO]
```

HTML (homepage.htm)

```
<!--#INCLUDE FILE="header.htm" -->
This is the main body.
</BODY>
</HTML>
```

HTML (header.htm)

```
<HTML>
<BODY BGCOLOR="BLUE">
```

When homepage.htm is loaded from Strategi, the line "…#INCLUDE…" line will be replaced by the contents of header.htm. You could add this #Include tag to any html files that you want to share the same header information (set background, title, etc), and then simply change header.htm when you want to change the look of the site.

The format of the #Include tag is as follows:

```
<!--#INCLUDE FILE="FilenamePlusOptionalURLData" →
```

The file is referenced in the same manner a file reference in a link.

For compatibility with other webservers, the word VIRTUAL may be used in the tag instead of FILE:

```
<!--#INCLUDE VIRTUAL="FilenamePlusOptionalURLData" →
```

There can also be HSM substitution within the #Include tag:

```
<!--#INCLUDE VIRTUAL="next.htm?item=<HSM NAME=itemvar ESCAPE=URL>" →
```

However, there must be no white space in the final URL.

The included text is passed direct to the browser - it is not HSM processed after inclusion. HSM processing of the included file uses variables passed by the #include, plus those set in the HSM file (if any) associated with the included file. The included file can use Subtitute_URL as normal.

Include files can include other files, etc, to a maximum depth of ten.

HSM Resource File Reference

File Construct

HSM Resource Files are identical in construct to .INI files. The file contains a series of "Groups" defining major functionality. Each group contains a series of KEYWORD=VALUE pairs which define the detail of that group's operation.

Unlike Windows .INI files, the groups are processed sequentially, so their sequence implies an order of operations.

Groups

Following are the valid groups and their purpose:

[SERVER REQUEST]

A request to an HSM server. The KEYWORD=VALUE pairs define field mappings and other information needed to make the request.

[REPLY]

A test for a particular reply from the immediately prior [SERVER REQUEST], and the associated processing, if matching.

[DO]

This group can be used to perform any actions (such as substitute URL) you do not want in another group type. Useful to control flow independently of REPLY, BLOCK, etc.

[BLOCK]

Controls the behavior of a block (defined by the <HSMBLOCK> tag in the template HTML).

[SUBMIT BUTTON]

Tests for a specific submit button, assuming the HSM resource file was called via a submit button in an HTML form. Its keywords define the associated processing, if matching.

Keywords and Values by Group

Following is a list of KEYWORD=VALUE pairs by group and their syntax:

[SERVER REQUEST]

A request to an HSM server. The KEYWORD=VALUE pairs define field mappings and other information needed to make the request.

```
SERVER=hsm_server_name
OPcode=server_OPcode_requested
Start_Length_Field=start_position,length,field_name - or - "literal"
```

Note that the first start_position is 1, not 0.

Length can be "10x20", indicating a 10-element array, with each element 20 characters long. Length can also be "5.1", indicating five digits with one decimal, zoned decimal format.

The HSM request buffer is first cleared to spaces, then named fields (if any) have their values moved to the specified positions. Character fields are truncated or space-filled to fit; numeric fields are converted to zoned decimal.

The OPcode is converted to uppercase before being sent.

You may use variables for server and OPcode names. If a server or OPcode name is preceded by an "&", then, if a variable with this name exists at that time in the HSM processing, the value for that variable will be used for the server or OPcode. If no such variable is defined, an error will result.

[REPLY]

A test for a particular reply from the immediately prior [SERVER REQUEST], and the associated processing, if matching.

```
OPcode=server_OPcode_replied
Start_Length_Field=start_in_buffer,field_length, field_name
```

Note that you can also use variables for OPcode names, as with the [SERVER REQUEST] group.

[DO]

This group can be used to perform any actions (such as substitute URL) you do not want in another group type. Useful to control flow independently of REPLY, BLOCK, etc.

```
Condition_Variable=field_name
Condition_Criteria=EQUAL -etc- (see below)
Condition_Compare_Value=field_name -or- "literal" - or - number
Substitute_URL=URL.htm
```

[BLOCK]

Controls the behavior of the corresponding <HSMBLOCK> tag in the template HTML.

```
Name=block_name
Condition_Variable=field_name
Condition_Criteria=EQUAL -etc- (see below)
Condition_Compare_Value=field_name -or- "literal" - or - number
Repeat_Count=field_name -or- number
```

[SUBMIT BUTTON]

Test for a specific submit button, assuming the HSM resource file was called via a submit button in an HTML form. Its keywords define the associated processing if matching the HTML "submit" type button name; each "submit" button on a HTML page has its own name:

```
Name=submit_button_name
```

The group would then contain general-purpose keywords such as Assign_Field_Value, Substitute_URL, etc., as described below.

Additional Keywords and Considerations

Conditions

As well as being used in blocks:

```
Condition_Variable=field_name, or HSM_Special_Value
Condition_Criteria=EQUAL -etc- (see below)
Condition_Compare_Value=field_name -or- "literal" - or- number
```

can be used to condition execution of all other groups:

- Server Request (immediately after OPcode)
- Reply (immediately after OPcode)
- Submit Button (immediately after Name)
- Do's

with the remainder of that group (and for a Server Request all of the following Replies) being ignored if the condition is false.

In a [REPLY], returned data values are not available for testing (because the Condition must be after the OPcode and ahead of any Start_Length_Field codes), but they can be tested in a second [REPLY] for the same OPcode.

A matching [REPLY] OPcode block does not prevent following blocks from being processed (although they would normally be for different OPcodes, and so do nothing), so one [REPLY] can retrieve the values and a following [REPLY] with the same OPcode can have conditional processing based on the values.

Conditions are evaluated when the Condition keywords are encountered, so BLOCK statements should generally be at the end of HSM processing, where variables will have their final values.

Condition Criteria

Based on just the Condition_Variable, the following conditions can be specified:

DEFINED	NOTDEFINED	
BLANK	NOTBLANK	NONBLANK
ZERO	NOTZERO	NONZERO

Based on the Condition_Variable and Condition_Compare_Value, these additional conditions can be specified:

EQUAL	NOTEQUAL	
GREATERTHAN	NOTGREATERTHAN	GREATERTHANOREQUAL
LESSTHAN	NOTLESSTHAN	LESSTHANOREQUAL
INGROUP	NOTINGROUP	

All fields contain text strings in HTML (defining numeric fields only affects how they are stored in the HSM buffer to/from the AS/400), and those that look like numbers will be compared as numbers, while those containing text will be compared as character strings.

Undefined variables are considered to be blank (thus the separate DEFINED test).

Note that submit buttons are, in fact, just variables, so:

```
[SUBMIT BUTTON]
Name=xyz
Assign_Field_Value=abc,"XYZ PRESSED"
```

is the same as

```
[DO]
Condition_Variable=xyz
Condition_Criteria=DEFINED
Assign_Field_Value=abc,"XYZ Pressed"
```

(the variable also has a value, as defined in the HTML).

The INGROUP and NOTINGROUP are used to check whether a specific specific Strategi User is a member of a Strategi Group. The Condition_Variable should be the user number, and the Condition_Compare_Value should be the name of a group.

```
[BLOCK]
Name=SHOWINFO
Condition_Variable=*USER.NUMBER
Condition_Criteria=INGROUP
Condition_Compare_Value="SALES"
```

Finally, note that the NON and NOT conditions are equivalent.

Explicitly Setting Field Values

In addition to values taken from the HSM buffer in a [REPLY], and from HTML form INPUT tags, a field can be explicitly assigned a value using:

```
Assign_Field_Value=field_name, field_name -or- number -or- "literal"
```

This takes effect immediately, overriding any Start_Length_Field in an earlier [REPLY]. The value may be changed by a following [REPLY] or Assign.

Assign_Field_Value can be used in all groups except BLOCK definitions.

Exporting Embedded Values

To store a field in the output HTML document, use:

```
Export_Field=field_name
```

This causes a hidden field to be embedded in the next HTML page. The field is placed in the HTML, immediately ahead of the </FORM> statement(s), as a hidden input field. The name of the hidden input field is the same as the name of the exported field, and the value is also the same. This means that

Array elements cannot be exported, only entire arrays and simple fields (Assign_Field_Value can be used to create a simple variable to export in place of an array element value).

Export_Field can be used in all groups except BLOCK definitions.

Array Elements

When a variable has been defined as an array, individual elements can be explicitly referred to in the HTML as "fieldname[n]" or "fieldname _ _ n" for element number "n", using 1 for the first element. In the HSM file, specific array elements can be set or retrieved.

```
[REPLY]
OPcode=GETARR
Start_Length_Field=1,3,X
Start_Length_Field=4,10x50,ARR
Assign_Field_Value=ARR[X],"SELECTED"
```

The two-underscore style is recommended in HTML tags, both HSM and INPUT, where the square brackets may in future have special HTML meaning.

For example, in the "two-underscore" style, the HTML would read as:

```
<INPUT TYPE=TEXT NAME=array__2>
```

and in the "square brackets" style, the tag would be:

```
<INPUT TYPE=TEXT NAME=array[2]>
```

Any fieldname ending with two underscores and a number will be treated as an array element.

Naming Conventions

Variable names can be up to 31 characters, must not start with a digit or have embedded spaces, and should be limited to A-Z, 0-9, and the _ (for future HTML compatibility).

Variable names, group names, keywords, and special values are case-independent (being converted to uppercase internally and in error messages).

Automatic HSM Start Position

It can sometimes be convenient in a [SERVER REQUEST] or [REPLY] group to simply specify a variables length in the Start_Length_Field, and not have to also specify the starting position in the data being sent or received. You can use an asterisk in Start_Length_Field to specify that the field should start immediately after the last field.

For example:

```
[SERVER REQUEST]
SERVER=STOCK
OPcode=GETDATA
Start_Length_Field=1,1,OP
```

```
Start_Length_Field=*,10,POSITIONING_KEY
```

Special Values

In place of a fixed OPcode for a [REPLY], *OTHER can be used, and will match any OPcode not yet matched in previous replies.

In place of a literal in Assign_Field_Value and Condition_Compare_Value, the following special values can be used:

*REPLY.OPCODE	The current reply OPcode (useful if OPcode=*OTHER was specified).
*USER.REFERER	The URL from which this request was made.
*CLIENT.IPADDRESS	The requesting client's IP address.
*CLIENT.USERAGENT	The HTTP browser identification.
*WEBSITE.IPADDRESS	The server-side IP address for the request.
*WEBSITE.NAME	The name of the website the request was made on.
*DATETIME	Date and time in the format: CCYYMMDDHHMMSS
*DATE	CCYYMMDD
*DATE.CCYY	CCYY
*DATE.CENTURY	CC
*DATE.YEAR	YY
*DATE.MONTH	MM
*DATE.DAY	DD
*TIME	HHMMSS (24hr clock)
*TIME.HOUR24	HH (24hr)
*TIME.HOUR12	HH (12hr clock)
*TIME.HOUR12_AMPM	'AM' or 'PM' uppercase english
*TIME.SECONDS	SS
*HOST.NAME	Value from RTVSYSVAL SYSNAME, eg. 'DEV400'
*HOST.OSVERSION	Value of DTAARA QSYS/QSS1MRI, eg. 'V4R2M0'
*HOST.SERIALNUMBER	Value from RTVSYSVAL QSRLNBR, eg. '1211221'
*HOST.COUNTRY	Value from RTVSYSVAL QCNTRYID 'US'
*HOST.MODEL	Value from RTVSYSVAL QMODEL, eg. 'S20' (leading spaces dropped)
*HOST.PROCESSOR	Value from RTVSYSVAL QPRCFEAT, eg. '2165'
*STRATEGI.VERSION	Strategi value RELEASE, eg. '132'
*STRATEGI.USERS	Strategi value USER#, eg.'000000235'

The following require the zone to have *BASIC authentication, else the value returned is blank:

*USER.NUMBER	When the zone being used requires authentication, this returns the number associated with the user logged in.
*USER.ACCESSNAME	The user's access name.
*USER.EMAIL	The value of the EMAIL field for the user
*USER.FIRSTNAME	The value of the FIRSTNAME field for the user
*USER.LASTNAME	The value of the LASTNAME field for the user
*USER.TITLE	The value of the TITLE field for the user
*USER.ORG	The value of the ORG field for the user
*USERATTR.*ua*	From the UA file (*ua* = the name of the Strategi User Attribute to retrieve).

An example of using these special values might be as follows:

```
[REPLY]
OPcode=NOSERVER
```

```
Substitute_URL=no_as400.htm

[REPLY]
OPcode=*OTHER
Assign_Field_Value=rpyOp,*REPLY.OPCODE
Substitute_URL=bad_op.htm
```

You can also use the special values in the html itself, by specifying the name in an HSM tag to equal some special value. For example:

```
<HSM NAME="*WEBSITE.NAME">
```

INDEX Special Values

Often when an HSMBLOCK is repeated it is useful to display the index for that particular repetition of the block. The *INDEX special value can be used to retrieve the index for a block. For example:

```
<TABLE>
<HSMBLOCK NAME=ITEMLOOP>
<TR><TD>
Number <HSM NAME=*INDEX>: <HSM NAME=ITEMNAME>
</TD></TR>
</HSMBLOCK>
</TABLE>
```

In this example, the first repetition of the block would replace *INDEX with one, the second with two, and so on.

It is also useful to use the special values *EVEN_INDEX and *ODD_INDEX. These values are unique in that they are used as the name of an HSMBLOCK. For example:

```
<TABLE>
<HSMBLOCK NAME=ITEMLOOP>
<TR><TD>
       <HSMBLOCK NAME=*EVEN_INDEX><FONT COLOR="BLUE"></HSMBLOCK>
       <HSMBLOCK NAME=*ODD_INDEX><FONT COLOR="RED"></HSMBLOCK>
Number <HSM NAME=*INDEX>: <HSM NAME=ITEMNAME>
</FONT>
</TD></TR>
</HSMBLOCK>
</TABLE>
```

The above example would alternate font color between red and blue.

HTTP Header Special Values

Often it is useful to have access to the information in the HTTP Header. HSM allows you to retrieve the HTTP Header information. The following special values used to retrieve such information:

```
*HTTPHEADER.<httpheader>   This will retrieve the specific corresponding value for
                           the HTTP Header keyword.
*HTTPHEADERLIST.KEYWORDS   Lists all keywords, each surrounded by '[ ]'
*HTTPHEADERLIST.VALUES     Lists all values, each surrounded by '[ ]'

*HTTPHEADERLIST.KEYWORDS_AND_VALUES
                           Lists all keywords and values, each surrounded by '[ ]'
```

Conditional [DO]

It is permissible, and very useful, to add conditions to a [DO] group. For example:

HTML (list.htm)

```
<HTML><BODY>
<FORM METHOD=POST ACTION="list_fw.hsm">
<SELECT NAME="list">
       <OPTION>Destination List
       <OPTION VALUE="product">Product List
       <OPTION VALUE="staff">Staff
</SELECT>
<INPUT NAME="menu" TYPE="SUBMIT" VALUE="Go">
</FORM>
</BODY></HTML>
```

HTML (list_fw.hsm)

```
[DO]
Condition_Variable=list
Condition_Criteria=EQUAL
Condition_Compare_Value="product"
Substitute_URL=product.htm

[DO]
Condition_Variable=list
Condition_Criteria=EQUAL
Condition_Compare_Value="staff"
Substitute_URL=staff.htm

[DO]
Substitute_URL=list.htm
```

The user will choose an option from a dropdown menu, which will, according to the value of the option they chose, run one of the DO groups, and will get to the Substitute_URL only if the conditions are met. If not, then the final DO will simply send them back to list.htm.

Switching to Another File

To link between files, use

```
Substitute_URL=URL_to_display_next.htm
Substitute_URL=nextURL
```

This causes processing to switch immediately to the specified file, which will be a .htm or .hsm on the same server (in which case all HSM variables are retained). Otherwise (if URL contains a ":", or is not a .htm/.html or .hsm), it becomes a normal HTTP redirection.

You can specify a literal value, such as "page2.htm", after the Substitute_URL, around which you must place quotation marks. You can also specify a variable name, such as nextURL, which should contain the exact value of the file to come next. Example:

```
Assign_Field_Value=nextURL,"page3.htm"
Substitute_URL=nextURL
```

The remainder of the current group is ignored after a Subtitute_URL, so when used this is generally the last line in a group.

Substitute_URL can be used in all groups except BLOCK definitions.

Values in Link URLs

All the above examples assume data comes from an HTML form, but values can also be included in the URL of a link. Multiple links to the same document might then have different effects depending on their data.

Values for HSM variables and submit buttons in the URL follow normal HTTP conventions, e.g.:

```
<A HREF=somedoc.htm?var=newval&sub=Submit+Button> link text </A>
```

would have the same effect as submitting:

```
<FORM METHOD=POST ACTION=somedoc.htm>
<INPUT NAME=var VALUE=newval>
<INPUT TYPE=SUBMIT NAME=sub VALUE="Submit Button">
</FORM>
```

Note the use of:

- ? after the real URL to introduce the values.
- & to separate the name=value pairs.
- + instead of embedded spaces.
- No spaces anywhere in the URL plus data combination.

In addition, special characters like %?+&=" in the names or values must be represented as :%xx", where xx is their hexadecimal ASCII value (a space would be "%20"). This is an HTTP requirement.

Whether a submit button value is needed depends on the logic in the HSM resource files, with something as little as:

```
somedoc.htm?var=newval
```

being often all that is needed, or for multiple variables:

```
somedoc.hsm?var1=newval1&var2=newval2
```

Notice that the URL can point to an .hsm file as well as an html file.

Returning HTTP Messages

HSM allows for forwarding to a standard HTTP message page, either as an HTTP normal response or as an HTTP forbidden message. This is useful for the integrity of look and feel for the website, as well as for use with non-browser clients (such as the Strategi command line HTTP utility).

The format for the HTTP redirect messages are as follows:

`RETURN_HTTP_FORBIDDEN=<code>,<text>`	Causes the HTTP 403 (Forbidden) to be sent with a structured HTML page that shows <code> and <text> in the normal error details.
`RETURN_HTTP_RESPONSE=<code>,<text>`	Causes the HTTP <code> to be sent with the <text> being the text for the HTTP response.

Code must be numeric, and can be either a literal or a variable. If it is a variable, the variable name must be preceded by an ampersand. The text can be either a literal or a variable, and if a variable it must also be preceded by an ampersand.

The keywords would be used in the same context as a Substitute_URL, in a REPLY or DO group:

```
[REPLY]
OPcode=HTTPERR
Start_Length_Field=1,10,CODE
Start_Length_Field=11,50,TEXT
RETURN_HTTP_RESPONSE=&CODE,&TEXT
```

Error Messages

Errors return HTTP error messages giving the HSM file and line number, using codes 523 and 524 to distinguish them from true server errors, such as code 500.

Error messages are delivered to the browser as normal documents, so all browsers will display the detail text, not just a standard string for the error number.

HTTP Error redirection through a URL

To redirect to a specific resource on a specific HTTP error, the following can be embedded in the URL:

```
*ERROR_HTTP_nnn=<url>?<url-data>
```

or alternatively,

```
*ERROR_SGI_nnn=<url>?<url-data>
```

This would be included in the HTML. For example, if you attempting to link to a parts description page that might not always exist, you could use the following in your HTML:

```
<A HREF="/parts/part0001.htm?*ERROR_HTTP_404=/parts/part_dft.htm">
```

Then, if "part0001.htm" did not exist, an error 404 would occur, and the user would be redirected to "part_dft.htm".

HSM File Upload & Pushfeed

Strategi Pushfeed

Strategi's Pushfeed allows a user to download files and printouts that have been sent to them *directly* from the Strategi webserver, rather than through the Strategi Java Applet. You can refer to the Strategi "Resources" area of your website for an example by going to "http://(your.as400.address)/resources" in your browser, and using the pull-down menu to select "Access Your PushFeed". You will be prompted to log in, and then you will see all files that have been sent to you with the SNDSGIF command, and all printouts that have been sent to you by having them placed in your Strategi Outq. See the Strategi Administrator's Manual for more information on sending files.

With Strategi's Pushfeed, each file has a specific path by which it is reachable through your browser. This path is in the format of:

```
HTTP://(your.as400.address)/(Website Zone)/*PUSHFEED/(Strategi User Number)/(File
Number)/(File Name)
```

The "Website Zone" refers to a Website Zone which uses basic authentication (See the Strategi Administrator's Manual for more information on setting up Zones). The user is therefore required to log in before they can access the file. If they have already logged into the Zone, they will not need to log in again. "Strategi User Number" is of course the number of the user logged in. "File Number" is the reference number of the pushed file, and "File Name" is the actual name of the file. An example of a full Pushfeed path would be:

```
HTTP://10.10.10.10/myzone/*PUSHFEED/000000699/0000007975/qtemp-test.dbf
```

Of course, you must have some way of retrieving all of this information. This is where the Push OPcodes for the Strategi System server are used. Refer to the "/resources/pushfeed" directory of your Strategi CD (version 1.5.3 or above) for the source of the pushfeed HTML and HSM on your website. If you do not have a Strategi CD version 1.5.3 or above, please visit the download area of BusinessLink's Support website at "support.businesslink.com", where you will be able to download the source as well.

Push OPcodes (For Strategi Pushfeed)

LSTPUSH

Returns several arrays containing information about files that have been sent to the logged in Strategi user. For LSTPUSH and DLTPUSH, the return OPcode and values are the same as those sent.

```
[SERVER REQUEST]
SERVER=*SYSTEM
OPcode=LSTPUSH
Start_Length_Field=1,9,*USER.NUMBER
Start_Length_Field=10,3,BGNSTATUS
* Beginning Status to include. "AVL"=Available, "XDL"=Deleted,
* "RTV"=Retrieved "APN=Pending, "ALL"=All Files
Start_Length_Field=13,3,ENDSTATUS
* Ending Status to include.
Start_Length_Field=16,10,REFPOS
* Reference Position, where in the file list to begin.
Start_Length_Field=26,1,SCROLLDIR
* Scrolling Direction, "F" for forward, "B" for back
Start_Length_Field=27,4.0,MAXENT
* The maximum number of entries to be returned

[REPLY]
OPcode= LSTPUSH
Start_Length_Field=1,4.0,NBRENT
* Number of entries returned
Start_Length_Field=5,1,MOREAFTER
* Y if more entries exist for scroll forward, else N
Start_Length_Field=6,1,MOREBEFORE
* Y if more entries exist for scroll back, else N
Start_Length_Field=7,20x10,FILREF
* File References
Start_Length_Field=207,20x9,USERS
* Users (currently all the same)
Start_Length_Field=387,20x3,STATUS
Start_Length_Field=447,20x1,PRIORITY
Start_Length_Field=467,20x4.0,NBROPN
* Number of times opened (retrieved)
Start_Length_Field=547,20x40,DESC
* Description
Start_Length_Field=1347,20x40,ORIGIN
Start_Length_Field=2147,20x64,FILENAME
Start_Length_Field=3427,20x1,TRANTYP,
* Transfer type
Start_Length_Field=3447,20x64,OVRMIM
* Overrdiding MIME type
Start_Length_Field=4727,20x10.0,SIZE
Start_Length_Field=4927,20x14,FILDAT
* File datetimestamp
Start_Length_Field=5207,20x14,CRTDAT
* Created datetimestamp
```

```
Start_Length_Field=5487,20x14,QUEDAT
* Queued datetimestamp
Start_Length_Field=5767,20x14,SNTDAT
* Sent datetimestamp
Start_Length_Field=6047,20x14,LSTDAT
* Last action datetimestamp
Start_Length_Field=6327,20x14,XDLDAT
* Deleted datetimestamp
```

DLTPUSH

Deletes the requested file.

```
[SERVER REQUEST]
SERVER=*SYSTEM
OPcode=DLTPUSH
Start_Length_Field=1,9,*USER.NUMBER
Start_Length_Field=10,4.0,NBRREF
* 0 Number of references supplied for deleteion (max 20)
Start_Length_Field=14,20x10,REFXDL
* Array of references to delete.
```

Note that the AS/400 does not keep a "created" time for IFS files - instead it keeps a status changed time, which starts off being create time but is affected by change of user authority etc. Windows on the other hand keeps the created time instead of status changed.

HSM File Upload

The Strategi webserver has the ability to upload files as well. From a page within a Strategi Zone that is secured with basic authentication (See the Strategi Administrator's Manual for more information on setting up Zones), one can select files from the drives of their PC, and send them to the AS/400 IFS. One can also use HSM to retrieve information about files on the IFS.

For example, imagine the file "postimg.htm" in the IMAGE zone, which is secured with BASIC authentication (for zone setup, reference the Strategi Administration Manual). Note that you can upload any sort of PC file you want, but images will serve as the example.

HTML (postimg.htm)

```
<HTML>
<FORM METHOD=POST ENCTYPE="multipart/form-data" ACTION="postimg.hsm"  >
New Image: <INPUT TYPE=FILE NAME="newfil">
<INPUT TYPE=submit VALUE="Upload!"><BR>
<BR>
<INPUT TYPE=HIDDEN NAME=newfil.authorityzone VALUE="IMAGE"><BR>
<BR>
</FORM>
</HTML>
```

HSM (postimg.hsm)

```
[DO]
```

The HSM file doesn't do anything at this point except indicate to the server that HSM processing must be done (retrieving the file). Setting the ENCTYPE attribute of the FORM tag to "multipart/form-data" allows the file download input type. The INPUT tag, which is set to type FILE, is the key piece, allowing the user to select a file for download. It should also be noted that "newfil.authorityzone" has been set to "IMAGE", meaning that the file will be secured in the same manner as files in the IMAGE zone.

Upon signing into the IMAGE zone, loading the page, selecting the file and clicking "Upload!", the selected file will be sent to the AS/400. The file will be saved with a handle, which is how it will be retrieved from then on. When a file is uploaded to the AS/400, it does not end up in the zone which contains the upload webpage. It is not even on the website portion of the IFS at all, and is not named the same name as the file originally was named. It is on the AS/400 IFS in the "/STRATEGI/publish" directory. The file will be named a ten digit number with no extension (this number is known as a "handle"), each file being one number larger than the previous uploaded file. For example, the first file uploaded will be named "0000000001", the second ""0000000002", etc. The "0000000001" file could then be accessed via webpage by referring to "*PUBLISH/0000000001". More likely, you would have a variable FILHDL (File Handle), and refer to it like this:

```
<IMG SRC="*PUBLISH/<HSM NAME=FILHDL>
```

How does one access the filehandle in the first place? Well, in the example HTML, the line <INPUT TYPE=FILE NAME="newfil"> means that when the form is submitted, the variable "newfil.handle" will be available within HSM. This newfil.handle could be then used on a following webpage, or stored to a database file via an HSM application, with other information as well. To access the handle information, you simply reference it in your HSM file.

HTML (postimg.htm)

```
<HTML>
<FORM METHOD=POST ENCTYPE="multipart/form-data" ACTION="postimg.hsm"  >
New Image: <INPUT TYPE=FILE NAME="newfil">
<INPUT TYPE=submit VALUE="Upload!"><BR>
<INPUT TYPE=HIDDEN NAME=newfil.authorityzone VALUE="IMAGE"><BR>
</FORM>
Last Image:
<IMG SRC="*PUBLISH/<HSM NAME=filhdl ESCAPE=URL>">
</HTML>
```

HSM (postimg.hsm)

```
[DO]
condition_variable=newfil.handle
condition_criteria=defined
assign_field_value=filhdl,newfil.handle
```

File Operation and Property OpCodes (For HSM File Upload)

In addition to the default file properties, you can add, change and retrieve properties of a file using OPcodes directed at the *SYSTEM HSM server. You can also move and copy files to areas of the website, as well as delete the uploaded files.

DELHFIL

Deletes the file and references to the file for the specified handle.

```
[SERVER REQUEST]
SERVER=*SYSTEM
OPcode=DELHFIL
Start_Length_Field=1,10,"*HANDLE"
Start_Length_Field=11,10,handle
...

[REPLY]
OPcode=DELHFIL
...
```

SETPROH

Creates a property for a given file when passed its handle. For SETPROH and CHGPROH, the return OPcode and values are the same as those sent, or ERROR if handle does not exist.

```
[SERVER REQUEST]
SERVER=*SYSTEM
OPcode=SETPROH
Assign_Field_Value=propname1,"Object Color"
Assign_Field_Value=propval1,"red"
Start_Length_Field=1,10,handle
Start_Length_Field=11,20,propname1
Start_Length_Field=21,128,propval1
```

CHGPROH

Changes the value of properties for a given file when passed its handle. If the property does not exist, it is created.

```
[SERVER REQUEST]
SERVER=*SYSTEM
OPcode=CHGPROH
Start_Length_Field=1,10,handle
Assign_Field_Value=propname1,"Object Color"
Assign_Field_Value=propval1,"green"
Start_Length_Field=11,20,propname1
Start_Length_Field=21,128,propval1
```

GETPROH

Retrieves the value of up to twenty properties for a given file when passed its handle and property requests.

```
[SERVER REQUEST]
SERVER=*SYSTEM
OPcode=GETPROH
Start_Length_Field=1,10,handle
Start_Length_Field=11,20,propname1
Start_Length_Field=31,20,propname2
Start_Length_Field=51,20,propname3
...

[REPLY]
OPcode=GETPROH
Start_Length_Field=1,10,handle
Start_Length_Field=11,128,propval1
Start_Length_Field=139,128,propval2
Start_Length_Field=267,128,propval3
...
```

FILEINFO

Retrieves information about a file on the IFS. The file can be specified via its location on the IFS, its location within the Website, or its handle.

```
[SERVER_REQUEST]
SERVER=*SYSTEM
OPCODE=FILEINFO
Start_Length_Field=1,10,"*IFS"
Start_Length_Field=11,1024,fullpath
* fullpath = complete IFS path, including leading "/"
```

or

```
[SERVER_REQUEST]
SERVER=*SYSTEM
OPCODE=FILEINFO
Start_Length_Field=1,10,"*WEBZONPTH"
Start_Length_Field=11,15,website
Start_Length_Field=26,15,zone
Start_Length_Field=41,1024,relpath
* relpath = relative path, no leading "/"
```

or

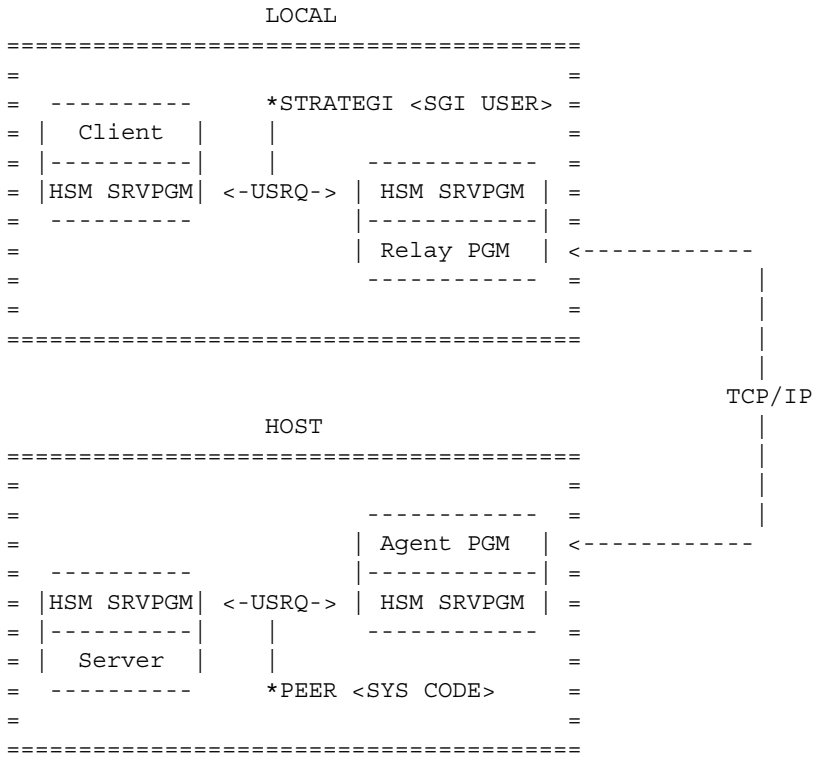
```
[SERVER_REQUEST]
SERVER=*SYSTEM
OPCODE=FILEINFO
Start_Length_Field=1,10,"*HANDLE"
Start_Length_Field=11,10,handle

[REPLY]
OPCODE=FILEINFO
Start_Length_Field=1,10.0,size
```

```
* Size of file in bytes.  If length is specified with number of decimals as above,
this size is left justified.  Else, it will have leading zeros.
Start_Length_Field=11,14,lastmod
* CCYYMMDDHHMMSS datetime (contents) last modified (st_mtime)
Start_Length_Field=25,14,lastchg
* CCYYMMDDHHMMSS datetime status changed (st_ctime)
Start_Length_Field=39,14,lastacc
* CCYYMMDDHHMMSS datetime last accessed (st_atime)
Start_Length_Field=64,11,objtyp
* Type of object, always "*STMF" for *HANDLE requests.
Start_Length_Field=75,5.0,codepage
Start_Length_Field=80,5.0,imgwdt
* Image width (if known for .gif .jpg .jpeg files, else zero)
Start_Length_Field=85,5.0,imghgt
* Image height (if known for .gif .jpg .jpeg files, else zero)
```

Distributed HSM Servers (DHSM)

DHSM (Distributed HSM) allows servers on peer installations of Strategi to be accessed. The system is considered a peer because the relationship between the systems is equal (that is there is not any implied hierarchy). For the purpose of this document the term "local" as applied to systems will mean the system which wishes to access an HSM server on another system. "Host" will mean the peer system on which the HSM server is being hosted.

The setup can be envisaged as:



DHSM Setup Steps

- Defining of peer systems (local and host systems)
- Creation of peer relay servers (local system)
- Enabling of peer services, and optionally (host system)
- Modification/addition of server authorities. (host system)

Licensing restrictions are not yet determined for DHSM, but all customers should be made aware that DHSM will be subject to license restrictions in the near future.

Authentication & Security
One of the key fields in the record is the serial number of the peer system. This must match RTVSYSVAL QSRLNBR; the Strategi software will retrieve the system serial number and pass it to the other end to identify itself. The serial number is analogous to a User's access name. Once the serial number is matched against the peer file, the defined system code is used in all other operations with reference to this system.

The two programs that connect for DHSM perform bi-lateral authentication with each other. Authentication requirements must be fulfilled at both ends before the connection can proceed with HSM transactions.

The fundamental basis for this is a 1024 bit dynamic exchange key. This key is analogous to a user's passphrase and is generated every time the record is "Reset" (Option 6 from the work-with). The key is hashed with several other peer and link specific pieces of information and the hash is sent for validation - this is a form of digest authentication. Thus, after the key is initially exchanged on the first connection with the peer, it is never resent.

Whenever a mismatch occurs with the exchange key the record is immediately locked and a reset is required. Whenever the record is reset, all connections are always accepted until the new challenge key is confirmed as stored at the other end (in practical term this is measurable in milliseconds from the time of TCP connection).

The peer definition can specify an IP filter from which connections will be accepted - this filter is checked by the both sides of the connection.

The peer definition can specify that SSL and, optionally, a client cert (SSL always uses a server cert) is required. The local side (which is making the TCP/IP connection) will validate it's recorded DNS name against the certificate common name, provided the DNS name is not an IP address. If a client certificate is used then the host side validates that the certificate received from the client matches the DNS name of the peer record (use of a client cert excludes configuring a simple IP address for the DNS name).

Setup Step 1: Define Peer System
The xxxSGIPEER commands are used to define a peer system. This definition is used to specify details of the peer system for making a connection (e.g. DNS), the requirements of the connection (e.g. SSL) and the status and identity of the system.

Setup Step 2: Configure Peer Relay Server
This uses CRTHSMSVR with a type of *PEER. The system name must be supplied, and the name of the server on the host system can also be entered, if different from the local server name. If the peer system serial number is the same as the local system, this indicates a locol-loop test server and an alternate server name must be entered. If it were not then the DHSM agent would end up making it's request of the original DHSM relay server, creating a deadlock.

Setup Step 3: Enabling Of Peer Services
The 3 Strategi values DHSMADDRESS, DHSMCERTIFICATE and DHSMTHREADS control the DHSM agent services. These need to be running on a host machine.

DHSMADDRESS specifies what TCP/IP address and ports to listen on. The Address can be *NONE (no DHSM incoming), *WEBSITES (use the addresses listed for websites, with different port), or a specific IP address. The Base Port is the non-ssl port to listen on, SSL is assumed to be this +1 and SSL+CTF this +2. The Use SSL flag enables SSL listening.

DHSMCERTIFICATE supplies the certificate for both client and server SSL connections. It may be *NONE or a website name. This is typically the principle website for the 400.

DHSMTHREADS controls the initial, increment and maximum threads DHSM can run. These are licensed and controlled just like HTTP Threads.

Setup Step 4: Modification of Server Authorities

DHSM servers making a request of the host server are checked as *PEER and the system code for the local system, as defined on the host system. Specific checking of individual users can only be done on the local system. The user details are not relevant or applicable on the host system for the purpose of checking server authority. For example, if the local system defined Strategi user 231, then this bears no relevance to the hosting system's user 231.

Any servers that are desired to be protected from access by a peer system should be changed to be restricted, and a *PEER *ALL *NO authority added.

Other Topics

Other Resources

Following are a few additional references of sample HTML/HSM source code.

SGIEXAMPLE

Strategi contains additional Strategi HSM skeleton server samples in the source file, `STRATEGI/SGIEXAMPLE *FILE`.

http://support.businesslink.com

The download area of the BusinessLink support website will contain the source for various demonstration / example applications.

Additional Questions

Listed below are a few questions and answers, which may help in solving a particular HSM programming need you may have:

Question:
On a reply, can data sent back to a page to a substitute URL, be made to check a checkbox in the HTML?

Answer:
To check a checkbox, output the strings "CHECKED" or " " as the value of an HSM reply variable, so the HSM [REPLY] code includes, for example:

```
Start_Length_Field=11,7,somevar
```

and then the HTML can be:

```
<INPUT TYPE=CHECKBOX NAME=somename <HSM NAME=somevar>>
```

Please note that this is "technically" invalid HTML, because the <HSM> tag has been nested inside the <INPUT> tag. This is acceptable, however, for Strategi's HSM, because the HSM server will have replaced the <HSM> tags before the document is delivered to the browser.

In the example above, we used different names for the form field NAME= and the HSM variable, so it was clear which was which, but in a real application would be natural for them to be the same.

Question:
On a reply, can data sent back to an HTML page (using HSM), be inserted into an input field, so it can be edited and sent back again?

Answer:
Data for the initial value of a textbox is just the value of the relevant reply variable, for example:

```
<INPUT TYPE=TEXT NAME=somename VALUE="<HSM NAME=somevar>">
```

with the quotes needed in case there are embedded spaces in the value.

Please note that this is "technically" invalid HTML, because the <HSM> tag has been nested inside the <INPUT> tag. This is acceptable, however, for Strategi's HSM, because the HSM server will have replaced the <HSM> tags before the document is delivered to the browser.

In the example above, we used different names for the form field NAME= and the HSM variable, so it was clear which was which, but in real life would be natural for them to be the same.

Appendix 1, Converting Old Servers

With Strategi version 1.3+ we now more tightly manage the interaction between HSM clients and servers to provide more flexibility and control of the HSM process. In addition, the ability to create host-based clients is now available.

While the process is very much "message queue" oriented, we now provide our own API calls to implement this, instead of using QRCVDTAQ and QSNDDTAQ directly. These API's are available as ILE bound calls and OPM program calls. XRCVDTAQ and XSNDDTAQ are provided for easy server modification to the new support, providing an identical "drop-in" replacement for the old data-queue calls.

The older, DTAQ based servers, continue to be supported exactly as before, but the direction would be clearly to migrate these servers to the new API's in the shortest possible time frame. Old servers *do not* inherit any benefits from the new design.

The example source file (now called SGIEXAMPLE) has been updated with tested and working servers and skeletons to illustrate the new options. It is *strongly* recommended that customers start writing new servers using the skeleton servers in SGIEXAMPLE.

A new include file, SGIINC, has been added and it contains HSM.H which has all public definitions required by a C program. The client side interface is currently for internal use only.

A new binding directory has been added, SGIBNDDIR, which contains all public Strategi bindable objects. For tested and working examples on how to create ILE-RPG programs with bound service program calls, refer to the SGIEXAMPLE/#MAKE source member.

Principle Benefits

- Simpler and more logical API interface specifically designed for HSM.
- Significantly reduced transaction overhead (2ms from 36ms on our 170).
- Multiple parallel servers.
- Strategi now detects and reports failures to reply to an OPcode.
- Increased transaction length, to 9999 bytes..
- Host-based (AS/400) clients.
- Ability for us to provide detailed usage/response statistics.

Existing Strategi HSM servers from v1.0.0 through v1.2.3 can be quickly converted to use the new HSM interface specification currently found in v1r3.

Once a server is converted it immediately inherits all the principle benefits outlines in the introduction for this document.

Converting the server to use the new HSM interface (as opposed to the data queue compatibility interface) provides the following additional benefits:

A simpler, cleaner and therefore a more "maintainable" program.
The client type for each transaction.
Full length client identification field (10 vs. 9).
Full length data (9999 vs. 9945).
Full length OPcode (10 vs. 8).
Reduced interface call overhead (by approximately 50%).

NOTE 1: The only non-mechanical change that may be required is if an old HSM server uses OPcodes, as part of the application, that begin with an asterisk. Any such OPcodes should be changed to ***not*** use an asterisk, in both client HTML and server code, before proceeding with any conversion. Such OPcodes would be blocked by the new interface, so must be changed. Strategi considers OPcodes beginning with an asterisk to be system OPcodes, like *STOP.

NOTE 2: A server no longer has any timeout, so any logic relating to timeout must be eliminated. It remains possible to implement periodic processing by creating a client program (submitted by the server on startup) that periodically submits a particular request to the server. The vast majority of existing servers should have no timeout processing, as data queue timeout cannot be relied on for periodic processing, due to no timeout occurring when the server is actively processing requests for clients.

COMPATABILITY INTERFACE Conversion

1. Check the existing HSM server for application OPcodes beginning with an asterisk; refer to note # 1, above.

2. Change the program parameter list.

   Original:

```
C      *ENTRY PLIST
C             PARM   DTQRPY 10
C             PARM   DTQRQS 10
C             PARM   DTQLIB 10
```

   Becomes:

```
C      *ENTRY PLIST
C             PARM   SVRNAM 10
```

3. Change the QRCVDTAQ call.

   Original:

```
C      CALL   'QRCVDTAQ'
C      PARM          DTQRQS
C      PARM          DTQLIB
C      PARM 1920     DTQLEN 50

C      PARM          RQSRCD
C      PARM -1       WAIT   50
```

   Becomes:

```
C      CALL   'XRCVDTAQ'
C      PARM ' '      DTQRQS  10
C      PARM ' '      DTQLIB 10
C      PARM 4150     DTQLEN 50
C      PARM          RQSRCD
C      PARM -1       WAIT   50
```

4. Change the QSNDDTAQ call.

   Original:

```
C      CALL 'QSNDDTAQ'
C      PARM          DTQRPY
C      PARM          DTQLIB
C      PARM 1920     DTQLEN
C      PARM          RPYRCD
```

   Becomes:

```
C      CALL   'XSNDDTAQ'
C      PARM ' '      DTQRPY 10
C      PARM ' '      DTQLIB 10
C      PARM 4150     DTQLEN 50
C      PARM          RPYRCD
```

5. Compile and run as previously.

ILE INTERFACE Conversion

1. Check the existing HSM server for application OPcodes beginning with an asterisk; refer to note # 1, above.

2. Convert the existing source to ILE, if necessary (`CVTRPGSRC`).

3. Change the message element definitions.

Original (as for OPM program):

```
I      DS
I              1      4150   RQSRCD
I              1      42     RQSHDR
I              1      10     RQSTYP
I              11     11     RQSF01
I              12     20     RQSUSR
I              21     42     RQSF02
I              43     50     RQSOPC
I              51     54     RQSLEN
I              55     4150   RQSD
I      DS
I              1      4150   RPYRCD
I              1      42     RPYHDR
I              43     50     RPYOPC
I              51     54     RPYLEN
I              55     4150   RPYD
```

Becomes:

```
D      RQSOPC S      10                          * REQUEST OPCODE
D      RQSD   S      1              DIM(9999)    * REQUEST DATA
D      RQSLEN S      4      P       0            * REQUEST DATA LENGTH
D      RQSCTY S      10                          * RQS CLIENT TYPE
D      RQSCID S      10                          * RQS CLIENT ID
D*
D      RPYOPC S      10                          * REPLY OPCODE
D      RPYD   S      1              DIM(9999)    * RPY DATA
D      RPYLEN S      4P     0                    * RPY DATA LENGTH
```

4. Change the program parameter list.

Original (as for OPM program):

```
C      *ENTRY PLIST
C             PARM   DTQRPY 10
C             PARM   DTQRQS 10
C             PARM   DTQLIB 10
```

Becomes:

```
C      *ENTRY PLIST
C             PARM   PSVR   10                  *SERVER NAME
```

5. Change the call to QRCVDTAQ (note CALLB in ILE program).

Original (as for OPM program):

```
C      CALL   'QRCVDTAQ'
C      PARM   DTQRQS
C      PARM   DTQLIB
C      PARM   1920   DTQLEN 50
C      PARM   RQSRCD
C      PARM   -1           WAIT   50
```

Becomes:

```
C      CALLB  'HSMRCVRQS'
C      PARM   RQSOPC *OPCODE
C      PARM   RQSDTA *DATA
C      PARM   RQSLEN *LENGTH
C      PARM   RQSCTY *CLIENT TYPE
C      PARM   RQSCID *CLIENT ID
```

6. Remove code to move RQSHDR to RPYHDR.

7. Remove any testing of RQSTYP or for data-queue timeout, and move the testing of the OPcode to the main line.

Original (as for OPM program):

```
C      DTQLEN CASEQ  0             DTQTMO
C      RQSTYP CASEQ  '*CLNT'       OPCODE
C      RQSTYP CASEQ  '*STOP'       ENDSVR
C             CAS                  BADRQS
C             END
C*
C* (TESTING OF OPCODES WAS USUALLY IN A SEPARATE SUBROUTINE)
C*
C*--------------------- STANDARD OPCODES
C      RQSOPC CASEQ  '*PING'       PNGSVR
C      RQSOPC CASEQ  '*STOP'       ENDSVR
C*--------------------- APPLICATION OPCODES
C      RQSOPC CASEQ  'GETLST'      GETLST
C      RQSOPC CASEQ  'GETITM'      GETITM
C      RQSOPC CASEQ  'GETLST2'     GETLS2
C      RQSOPC CASEQ  'GETITM2'     GETIT2
C*--------------------- OPCODE NOT RECOGNIZED
C             CAS          BADRQS
C             END
```

Becomes:

```
C*--------------------- STANDARD OPCODES
C      RQSOPC CASEQ  '*PING'       OPPING
C      RQSOPC CASEQ  '*STOP'       OPSTOP
C*--------------------- APPLICATION OPCODES
C      RQSOPC CASEQ  'GETLST'      GETLST
C      RQSOPC CASEQ  'GETITM'      GETITM
C      RQSOPC CASEQ  'GETLST2'     GETLS2
C      RQSOPC CASEQ  'GETITM2'     GETIT2
C*--------------------- OPCODE NOT RECOGNIZED
C             CAS          BADRQS
C             END
```

8. Change the QSNDDTAQ call.

Original (as for OPM program):

```
C      CALL   'QSNDDTAQ'
C      PARM          DTQRPY
C      PARM          DTQLIB
C      PARM 1920     DTQLEN
C      PARM          RPYRCD
```

Becomes:

```
C      CALLB  'HSMSNDRPY'
C      PARM          RPYOPC *OPCODE
C      PARM          RPYDTA *DATA
C      PARM          RPYLEN *LENGTH
```

9. Remove any subroutines dealing with data queue timeout; if periodic processing is required, refer to note # 2, above.

10. Add subroutine for processing `*STOP` by replying with *STOPPED and exiting. Program structure for this can be found in the Strategi library, Source file `SGIEXAMPLE`, Member `HSMSKLRPG`.

For Example:

```
C*STOP REQUEST - Server Responds With '*Stopped' And Flags
C*Need To Exit
C*
C      HSMOPCSTOP           BEGSR
C*
C      RQSCTY IFNE          '*CONTROL'
C             EXSR          HSMOPCERR
C             ELSE
C             MOVEL(P)      '*STOPPED'    RPYOPC
C             Z-ADD  0      RPYLEN
C             ENDIF
C             SETON         LR
C             ENDSR
C*
```

11. Compile program using `CRTBNDRPG` or `CRTRPGMOD/CRTPGM`, specifying `BNDDIR(STRATEGI/SGIBNDDIR)`.

Appendix 2, Additional HSM Administrative Features

HSM Authorities

An HSM server can be secured using HSM authorities. HSM authorities define restrictions and permissions for Strategi Users, Strategi Groups, or even specific AS/400 users (through the *HOSTUSER specification of the Strategi User), with regards to whether they can use certain OPcodes. From the "Work with HSM Servers" (WRKHSMSVR) menu option, you can work with authorities for any server.

In order for authority settings to be checked, the HSM server must have restriction turned on, via the RESTRICT parameter (CRTHSMSVR, CHGHSMSVR). In order for authority checking to have user information to check against, the HSM application's web pages must reside in an authenticated zone.

HSM Performance Monitoring

HSM Servers can be monitored for performance information. A server will not be monitored unless the performance monitoring parameter, PFRMON, is set to *YES.

Performance monitoring tracks time, in milliseconds, between HSM requests and replies. Minimums, maximums, averages, and a total count are recorded. This information can be retrieved with the DSPHSMPFR command.

Customized Login Pages

Strategi website zones can be configured to allow *CUSTOM authentication. This allows you to create customized login webpages, rather than using the browser's default login dialogue (as with *BASIC).

With *CUSTOM authentication, one page within the *CUSTOM authenticated zone will be delivered without any authentication whatsoever (although SSL requirements will apply where configured). The name of this page is specified by the LOGINURL Strategi value ("login", for example, is the default). This can be specified with or without an extension, depending upon whether you wanted different login page types delivered based on zone (One zone might deliver "login.htm", while another "login.shtm"). This page will be used to log the user into the zone.

The login page must contain a form which makes a post, the action of which is another webpage within the zone, specifying "?*LOGIN" after the name of this file. Also within the form must be two input fields, "*LOGINUSR", and "*LOGINPWD". These fields will be where the user signs in. An example of a valid login webpage follows:

HTML (login.htm)

```
…
<Form Action="main.htm?*LOGIN" Method="POST">
User Name <Input Type="text" Name="*LOGINUSR"><BR>
Password <Input Type="password" Name="*LOGINPWD" value=""><BR>
<Input Type=Submit Value="LOGIN">
</Form>
…
```

Main.htm file, in turn, would usually contain links that allow the user to log off.

HTML (Main.htm)

```
…
You have reached a page using HTTP Custom Authentication
<A HREF="login">LOGOUT</A> <br>
<A HREF=".?*SESSION=LOGOUT&*LOGOUTREDIRECT=/homepage.htm">LOGOUT AND GO HOME</A>
<BR>
…
```

These logout links can be included on any page within the zone. The logout link has a very specific structure. First, notice that no actual html page (just a "." Instead) is being referenced. Instead, two variable pairs are being sent. *SESSION=LOGOUT tells the server to log the user out of the zone, and *LOGOUTREDIRECT sends them to a different webpage. The fact that the redirect is to "/homepage.htm" means that it will be passed to the homepage.htm in the root of the website. Finally, notice that returning to the login page effectively logs the user out.

HSM Server Packaging & Installation Commands

The PKGHSMSVR and INSHSMSVR are Strategi commands that allow HSM developers to distribute HSM applications to other AS/400's running Strategi. PKGHSMSVR is used to prepare a HSM application for distribution on the source AS/400 while INSHSMSVR is used to install the HSM application on the destination AS/400. A simple explanation of the command parameters are listed below. For more detailed information, refer to the online help associated with each command parameter.

PKGHSMSVR	SVRNAM	name of server to package
	LIB	library where server program resides
	DIR	website code and subdirectory of IFS objects
	DEV	AS/400 storage device to save to (*DEVD or *SAVF)
SAVF	name of save file to contain objects for distribution	
	PGM	packaging program (default HSMPKGEZ)

Implementation Notes

- A data area of type *CHAR length 10 named 'TGTRLS' must reside in the library specified on the command. This data area contains the target release that the objects will be saved at. The format of the data area contents are of form VxRxMx (ie V4R2M0).
- All objects must be owned by a profile that the user who installs the application on the destination AS/400 has authority to. If this is unknown, it is recommended that all objects be owned by Strategi user profile SGIOBJOWN. This profile has the highest probability of existing on both target and destination AS/400s.
- The packaging program HSMPKGEZ is the default-issue packaging program used by the command. A different packaging program may be specified if it exists. For an example of the procedures that this program performs, refer to the source member HSMPKGEZ in file SGIEXAMPLE in the Strategi library.
- From a development standpoint it is recommended that the target AS/400 OS/400 version be confirmed before the HSM application is packaged. This will determine the target release used in the package process and avoid complications on the installation. Failure to do so can force a repeat of the package process and delay installation time.

INSHSMSVR	SVRNAM	name of server to install
	LIB	library for server objects
	DIR	website code and subdirectory of IFS objects
	DEV	AS/400 storage device to restore from (*DEVD or *SAVF)
	SAVF	name of save file containing objects to install
	SVROPT	server overwrite option
	LIBOPT	library overwrite option
	DIROPT	directory overwrite option

Implementation Notes

- If a normal completion message is not received, display the job log to obtain further detail on the possible problem. The most likely causes are security format changes due to insufficient authority. This is usually resolvable with operator intervention (i.e. granting proper authorities) and executing the command again but may require a repackage on the source system depending on the severity.

Index

****

*PGM 16
*PING 15, 17
*SITE_DIR 22
*STOP 15, 17

[

[BLOCK] 29, 31
[DO] 29, 31
[REPLY] 29, 30, 32, 51
[SERVER REQUEST] 29, 30
[SUBMIT BUTTON] 30, 31

<

<HSM NAME> 26
<HSMBLOCK> 26, 29
<INPUT TYPE=> 51

5

5250 interface 11

A

Array elements 34
ASCII 17
Assign_Field_Value 33
ATFM19G 51

C

CHGPROH 42, 43, 46
Client 17
Client/Server 13
Communications programming 11
COMPATABILITY INTERFACE conversion 54
Condional [DO] 36, 37
Condition criteria 32
Conditions 32
Converting old servers 53, 59
Converting older HSM servers 53

D

Data queue 13
DataBlock 17

E

EBCDIC 17
Embedded values 33

Error messages....34, 39
Examples
Guest book....23
Stock inquiry....24
Visitor counter....21
Export_Field....34

F

Field values....33
File properties....41, 42, 44, 46
FILEINFO....47

G

GETITEM....14, 15, 18
GETLIST....15, 18
GETPROH....46
Green screen....11
Guest book....23

H

HSM groups
[BLOCK]....29
[DO]....29
[REPLY]....29
[SERVER REQUEST]....29
[SUBMIT BUTTON]....30
HSM keywords
[BLOCK]....31
[DO]....31
[REPLY]....30
[SERVER REQUEST]....30
[SUBMIT BUTTON]....31
HSM Resource File....29
HSMReply....14
HSMRequest....14
HTML....21
HTML client....13
HTML Keywords
<HSM BLOCK>....26
<HSM NAME>....26

I

ILE INTERFACE Conversion....56
INI....29

L

Link URL values....38

M

Message elements....18

N

Name conventions 34, 39

R

RPG 13

S

SAMPLE 51
Screen scraper 11
Server 17
Server design 16
Server interface calls 19
Server logic 15
SETPROH 46
Special values 35, 37
Start_Length_Field 51
Stock inquiry 24
Strategi 11
Substitute_URL 38
Switching files 37
System requirements 12

V

Visitor counter 21

W

Work with HSM servers 16