BusinessLink Software Support

Strategi

HSM Programmer's Guide

Version V2R1

This manual applies to Strategi version V2R1 and later and was last revised in July, 2004.

Printed in the United States of America.

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.

With some installs of Adobe Acrobat, your printer may not resolve the characters correctly, and once printed, all characters will appear as 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 us via email at support@businesslink.com.

Sincerely,

BusinessLink Software Support

Contents

Chapter 1. Introduction

Strategi High Speed Messaging

Strategi High Speed Messaging (HSM) is a groundbreaking communications technology that allows a high degree of separation between user interface (web page look-and-feel) and processing (programming tasks). No longer must a Graphics Designer become a programmer in order to have dynamic web page functionality. 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," HSM blends seamlessly with Strategi to connect remote clients to your iSeries over the Internet.

HSM's Relevance to Your Business

HSM 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 HSM makes it possible to produce striking graphical applications that communicate effortlessly with your iSeries.

HSM utilizes the iSeries 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 HSM allows authorized users to converse with your iSeries via the Internet from a web browser without actually letting the user onto your iSeries.

How? HSM acts as a "middleman" between the Internet and your iSeries, limiting the user’s interaction with the iSeries 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. 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 iSeries program. A web page on your Strategi-hosted web site makes a request to the Strategi subsystem on the iSeries. Almost instantly, Strategi relays the message to your HSM application on the iSeries. Then, your HSM application processes your request and calls the reply API. Strategi handles relaying the iSeries program's reply back to the next web page.

HSM System Requirements

iSeries Requirements

- OS/400 V4R2M0 or later running Strategi.
- LAN connections as required by Strategi.
- Approximately 60 MB of iSeries disk space to run Strategi.

PC Requirements

- Web browser

Chapter 2. Understanding HSM

A Practical Example

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

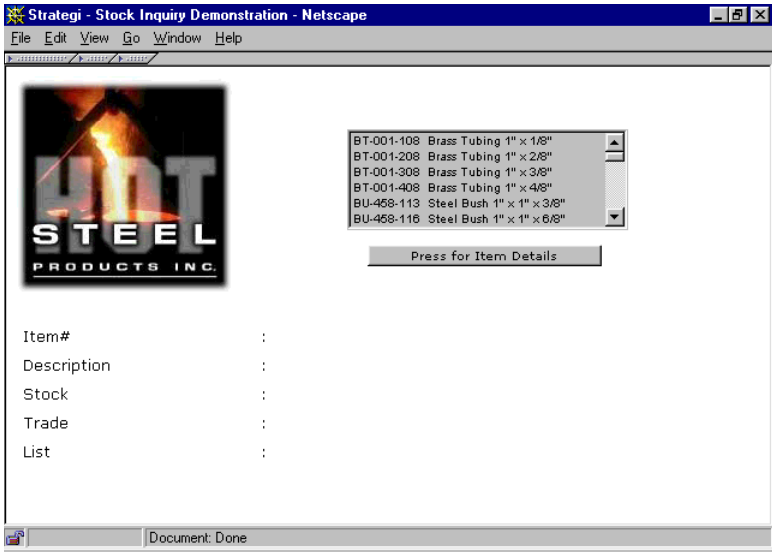
The software client is a web browser that makes HTTP requests for web pages. The web pages are written using standard HTML and HSM Resource files. The "client" has no specific knowledge of the server logic or the iSeries; Strategi handles all web page communications in the background.

The "Server" can be written in RPG, CL or any other programming language on the iSeries. The server is loaded, run and managed by Strategi. It has no knowledge of how it connects to clients or even how client requests get sent to it; Strategi handles managing the request. The server simply implements a Strategi API to receive the client request. Furthermore, it has no knowledge of how clients get the replies it sends using the Strategi API. How the correct reply gets back to the correct client is even outside the server's knowledge. Once again, Strategi handles sending and receiving information to and from the client so the server can focus on solving the business problem not communications infrastructure.

Client/Server Interaction

The following demonstrates a typical client/server interaction using HTML and a web browser as the client. Some detail is purposely omitted to simplify the explanation and provide a feel for HSM interaction, rather than detailed program logic or client HTML programming.

After going to the desired web page, the main form is displayed,


Strategi - Stock Inquiry Demonstration - Netscape
File Edit View Go Window Help
HOT STEEL PRODUCTS INC.
Press for Item Details
Document: Done

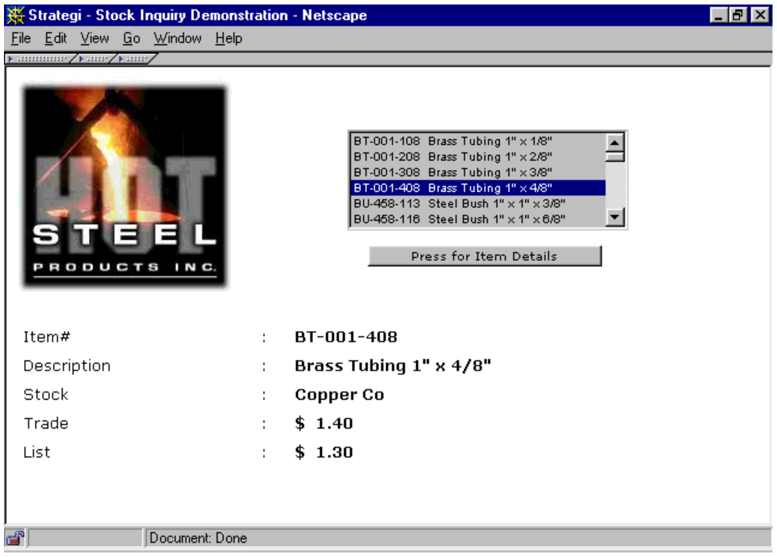
```
BT-001-108  Brass Tubing 1" x 1/8"
BT-001-208  Brass Tubing 1" x 2/8"
BT-001-308  Brass Tubing 1" x 3/8"
BT-001-408  Brass Tubing 1" x 4/8"
BU-458-113  Steel Bush 1" x 1" x 3/8"
BU-458-116  Steel Bush 1" x 1" x 6/8"
```

Item#	:
Description	:
Stock	:
Trade	:
List	:

The user selects an item in the list and clicks the "Details" button - this results in an HTTP request to the Strategi web server to process information contained in the HTML form and HSM Resource file involved. The HSM Resource file generates "resources," such as the HSM Server name to pass the HTML form information details to.

The item number selected is passed as a field in the DataBlock to the awaiting server. The server returns a reply DataBlock using a Strategi API, which is then parsed into fields by Strategi using the HSM Resource file information. The fields are then substituted into the HTML template for display in the list box (see example below). This step could be repeated several times as the user examines pricing and availability for different items.

Upon examination of the HTML source for any web page returned with iSeries data in it by the Strategi web server, and compare it with the raw .HTM file, you can see how seamlessly Strategi inserts the correct data from the reply DataBlock into the HTML.


Strategi - Stock Inquiry Demonstration - Netscape
File Edit View Go Window Help
STEEL PRODUCTS INC.
Press for Item Details
Document: Done

```
BT-001-108  Brass Tubing 1" x 1/8"
BT-001-208  Brass Tubing 1" x 2/8"
BT-001-308  Brass Tubing 1" x 3/8"
BT-001-408  Brass Tubing 1" x 4/8"
BU-458-113  Steel Bush 1" x 1" x 3/8"
BU-458-116  Steel Bush 1" x 1" x 6/8"
```

Item#	:	BT-001-408
Description	:	Brass Tubing 1" x 4/8"
Stock	:	Copper Co
Trade	:	$ 1.40
List	:	$ 1.30

In this example, all data, including the list box of items and the pricing values, are coming live from the iSeries. No data is stored locally on the PC, and all pricing logic is retained on the iSeries; no re-coding in an unfamiliar PC language. The end-user has no obvious indication data is actually coming from the iSeries as opposed to the PC, since performance is similar to PC-based applications.

Server Logic

The server logic for processing a request for a list of items, or details for one item, can be pseudo-coded as:

```
Do
Receive Client Request
If Request is "GETLIST"
    Read List of Items from Item Master File
    Put item list (array) into reply structure
    End If
If Request is "GETITEM"
    Validate Item number passed from Client
    Read Item Warehouse record to get stock, $, etc.
```

```
      Put item details into reply structure
      End If
  [.. test for and process *STOP and *PING requests ..]
  Send Client 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.

HSM Processing

Strategi's built-in web server delivers static as well as dynamic, content-variable web pages. A static web page does not require variable substitution prior to final rendering. A dynamic web page contains variable content – content from an application interacting with the Strategi environment, or by utilizing Strategi's built-in HSM Special Values.

Invoking HSM Processing

There are two methods for invoking HSM processing. The first method involves using an HSM Resource file whose file name is the same as the requested resource (e.g., homepage.htm), but has the .HSM extension (e.g., homepage.hsm). The second method involves using a file extension known to Strategi to always invoke HSM processing, .SHTM and .SHTML. The latter file types refer to Include processing, also known as Server Side Includes.

Once HSM processing is invoked for the HTTP request, resources generated are available for the final rendered document, regardless of whether or not a corresponding .HSM file exists for it. A common example usage of this feature is an "Error" HTML page that includes HSM variable substitution of the error information. Complex HSM applications that make numerous server requests benefit by having a single error page that is substituted when a particular error occurs, and provides a consistent user-interface experience with ease of maintenance. By having error information substituted on the HTML page it provides the choice of displaying the data on the web page or including it as "hidden" text. This way it can be hidden from direct view, but accessible if viewing the HTML source rendered in the browser.

Strategi performing HSM processing based upon specific criteria eliminates unnecessary processing and ensures HSM processing is done according to application design. By default, Strategi does not perform HSM processing for an HTTP request, it must be specifically requested via one of the two methods described.

Example HSM Processing Invocation

When an HTTP request is made for HOMEPAGE.HTM, Strategi looks for the file HOMEPAGE.HSM. If HOMEPAGE.HSM exists, Strategi initiates HSM processing and processes HOMEPAGE.HSM from top-to-bottom until reaching the end of the file, or is directed to get another file and process it. Once Strategi completes processing HOMEPAGE.HSM and without being directed to process another file, Strategi performs the variable substitutions for HOMEPAGE.HTM and delivers the rendered document to the client.

In many instances the client is a web browser capable of reading and displaying HTML pages. However, Strategi supports HSM Clients that make requests to HSM Servers. The Strategi HSM APIs that support HSM Client programs are located in the chapter on HSM Client Design.

See the section Server Side Includes and chapter HSM Resource File Reference for detailed HSM processing example usage.

Chapter 3. HSM Resource File Reference

The HSM Resource file, commonly referred to as an HSM file, is used to generate resources such as variables and block conditions that can be used within the final document, or to control the content of the final document. The HSM file also makes host resources (such as requests to HSM servers) available to the document author and triggers alternative document retrieval (e.g., server-side includes).

Once a .HSM file is encountered, either by explicit reference via an HTTP request or substitute_url, or implicitly due to requesting the corresponding, same-named file but different extension, HSM processing is invoked.

The following sections define the structure and layout of the HSM Resource file, including easy-to-understand examples demonstrating use and outcome.

File Construct

The HSM Resource File contains a series of "Groups" defining major groupings of functionality. Each group contains a series of KEYWORD=VALUE pairs that define the detail of that group's operation.

Each group contains a "header" to differentiate one group from another. All information following the group heading is associated with that group until another heading is encountered. Strategi reads the HSM file from top-to-bottom and processes the groups sequentially. This makes group ordering within the file important and ensures the desired logical flow and processing is adhered to.

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 '_', for future HTML compatibility.

Variable names, group names, and keywords are case-insensitive, being converted to uppercase internally and in error messages.

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

Adding comments to the HSM Resource file is allowed, and encouraged for readability reasons when others need to make file modifications. Begin the line with either the '*' or ';' character to indicate a comment.

Blank lines have no affect upon processing and serve only to provide an "intelligent use of white space."

Variable Creation

Working with variables during HTTP requests is a significant benefit provided with Strategi's HSM component.

Each HTTP request starts out with no variables defined. When a variable is defined, its name is placed in the list of variable names known to the web server for this request. Once a variable is defined, it remains defined until the end of the HTTP request, at which time the entire variable list is destroyed.

See the condition_criteria section for additional information on testing variable criteria.

Group Definitions

HSM Resource file "Groups" are comprised of a series of block '[group-header]' groupings to define specific KEYWORD=VALUE pair processing logic. The KEYWORD=VALUE pairs are associated with the immediately preceding block. Because Strategi reads the HSM file from top-to-bottom, when the next block is encountered, processing for the current block is considered complete.

There are five HSM groups that define HSM processing directives: **[BLOCK]**, **[DO]**, **[REPLY]**, **[SERVER REQUEST]**, and **[SUBMIT BUTTON]**.

[BLOCK]

The **[BLOCK]** group notifies Strategi the KEYWORD=VALUE pairs following determine the true/false nature of the <HSMBLOCK> </HSMBLOCK> tag grouping in the HTML file. The **name** defined in the [BLOCK] group corresponds to the **name** attribute in the <HSMBLOCK> tag in a related HTML file, and therefore, controls the behavior of the <HSMBLOCK> </HSMBLOCK> tag grouping.

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

Group

```
[BLOCK]
```

Required Keywords

```
name
```

Required Keyword Order

```
First position in group: name
```

Examples

```
**************************
* Example Block Group
**************************
[BLOCK]
name= SHOWBLOCK
condition_variable= ShowList
condition_criteria= EQUAL
```

```
condition_compare_value= "Y"
```

[DO]

The **[DO]** group notifies Strategi the KEYWORD=VALUE pairs following do not pertain to any other group, but still must be processed. This group is useful to control processing flow independently of other groups, such as **[REPLY]** and **[BLOCK]**. The **[DO]** group can be conditionally processed using the Condition_xx keywords, and is particularly useful in initializing variables prior to other group actions.

In early versions of Strategi an empty **[DO]** group was required to initiate HSM processing for an HTML page that only required HSM Special Value use.

Group

```
[DO]
```

Required Keywords

```
(none)
```

Required Keyword Order

```
(not applicable)
```

Examples

```
**************************
* Example Do Group
**************************
[DO]
assign_field_value= VARSVR, "MYSERVER"
```

[REPLY]

The **[REPLY]** group notifies Strategi the KEYWORD=VALUE pairs following pertain to the immediately prior **[SERVER REQUEST]**. If keyword Opcode matches the reply opcode sent by the HSM server, then and only then will the remaining KEYWORD=VALUE pairs for that particular **[REPLY]** block be processed.

Group

```
[REPLY]
```

Required Keywords

```
opcode
```

Required Keyword Order

```
First position in group: opcode
```

Examples

```
**************************
* Example Reply Group
**************************
[REPLY]
opcode= GETDETAILS
start_length_field=   1,   40, PartDescription
start_length_field=   *,   10, UnitOfMeasure
```

```
start_length_field=   *, 11.2, UnitCost
start_length_field=   *, 11.2, UnitPrice
```

[SERVER REQUEST]

The **[SERVER REQUEST]** group notifies Strategi the KEYWORD=VALUE pairs following pertain specifically to making a request, and optionally, passing data to the HSM server defined by the keyword Server. The server name specified for Server must exist or an error will occur.

In a **[REPLY]**, returned data values are not available for testing because the Condition must be after the **opcode** and before any **start_length_field** codes. The returned data values 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.

Group

```
[SERVER REQUEST]
```

Required Keywords

```
server
opcode
```

Required Keyword Order

```
First position in group: server
Second position in group: opcode
```

Examples

```
*******************************
* Example Server Request Group
*******************************
[SERVER REQUEST]
server= MYSERVER
opcode= GETLIST
start_length_field=   *,  15, PartNumber
```

[SUBMIT BUTTON]

The **[SUBMIT BUTTON]** group notifies Strategi the KEYWORD=VALUE pairs following require processing based upon the submit button clicked in an HTML form. The **[SUBMIT BUTTON]** group allows specific processing to occur based solely upon the invoking submit button. The **[SUBMIT BUTTON]** group is a test to determine which submit button was clicked.

Group

```
[SUBMIT BUTTON]
```

Required Keywords

```
name
```

Required Keyword Order

```
First position in group: name
```

Examples

```
*****************************
* Example Submit Button Group
*****************************
[SUBMIT BUTTON]
name= UPDATE
assign_field_value= ButtonClicked, "UPDATE"

[SUBMIT BUTTON]
name= DELETE
assign_field_value= ButtonClicked, "DELETE"
```

Keyword Definitions

One of the powerful aspects of HSM is the ability to pass variables from one HTML page to another. Most of the keywords pertain to variable manipulation, and therefore, the VALUE component of the KEYWORD=VALUE pair usually contains a variable name.

KEYWORD is a case insensitive Strategi-known instruction, usually describing how to manipulate a variable made to the Strategi web server from an HSM file. The majority of keywords pertain to variable manipulation. The non-variable manipulation keywords relate to other processing instructions, such as defining an HSM server for a request or making an HTTP redirection.

VALUE syntax is dependent upon KEYWORD. Some KEYWORDS require a single value for VALUE (e.g., substitute_url=somepage.htm), while others require multiple comma delimited values (e.g., start_length_field=1,25,ItemNumber).

assign_field_value

assign_field_value allows for explicitly assigning a value to a field (variable). The assignment takes place immediately, overriding any previously set value for the field.

Keyword

```
assign_field_value
```

Value

```
field-name, field-name | number | literal
```

Allowable Groups

```
[DO]  [REPLY]  [SERVER REQUEST]  [SUBMIT BUTTON]
```

Examples

```
[DO]
assign_field_value= MaxNumberToReturn, 10
```

…results in assigning the variable MaxNumberToReturn the numeric value 10.

```
[REPLY]
opcode= *OTHER
assign_field_value= UnknownCode, *REPLY.OPCODE
```

```
substitute_url= unknownOpcode.htm
```

…results in conditionally assigning the variable UnknownCode the value of the HSM Special Value *REPLY.OPCODE variable.

```
[SUBMIT BUTTON]
name= UPDATE
assign_field_value= ButtonClicked, "UPDATE"
```

…results in conditionally assigning the variable ButtonClicked the literal value 'UPDATE' when the submit button UPDATE from an HTML form is clicked.

Condition Keywords

The Condition set of HSM keywords are inter-dependent and dynamically determine group execution. The keywords are **condition_variable**, **condition_criteria** and **condition_compare_value**. Based upon their inter-dependence values, a group may or may not be processed, hence the word "Condition" used in the keywords. If the specific criteria for processing that group is met, the remaining keywords in that group are processed. If the criteria is not met, all subsequent keywords in that group are bypassed and processing continues with the next group.

condition_variable

condition_variable is used to identify the variable for comparison purposes. Used in conjunction with keywords **condition_criteria** and **condition_compare_value**.

Keyword

```
condition_variable
```

Value

```
field-name
```

Allowable Groups

```
[DO]   [BLOCK]   [REPLY]   [SERVER REQUEST]   [SUBMIT BUTTON]
```

Examples

```
[DO]
condition_variable= MoreForward
condition_criteria= EQUAL
condition_compare_value= "Y"
assign_field_value= Direction, "F"
```

results in variable Direction being assigned the value 'F' provided MoreForward is equal to 'Y'. If it does not, Direction will contain the value it had prior to the **[DO]** group, if defined.

```
[DO]
condition_variable= MoreForward
condition_criteria= EQUAL
condition_compare_value= Yes
assign_field_value= Direction, "F"
```

results in variable Direction being assigned the value 'F' provided MoreForward is equal to the value for variable Yes. If it does not, Direction will contain the value it had prior to the **[DO]** group, if defined.

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETDETAILS
condition_variable= PartNumber
condition_criteria= DEFINED
start_length_field=   1,  15, PartNumber
```

results in the server request being executed provided PartNumber is defined. If it is not, the server request and corresponding **[REPLY]** group are ignored.

```
[BLOCK]
name= SHOWINFO
condition_variable= *USER.NUMBER
condition_criteria= INGROUP
condition_compare_value= "SALES"
```

results in the corresponding HTML section to display its contents provided *USER.NUMBER is a member of the SALES Strategi group. If it is not, the section defined will be removed from the HTML template before final rendering occurs.

```
[BLOCK]
name= SHOWBLOCK
condition_variable= ShowList
condition_criteria= EQUAL
condition_compare_value= "Y"
```

results in the corresponding HTML section to display its contents provided ShowList is equal to 'Y'. If it does not, the section defined will be removed from the HTML template before final rendering occurs.

condition_criteria

condition_criteria is used to condition the criteria used for comparing a field to a value. Used in conjunction with keyword condition_variable, and depending upon the **condition_criteria** value, keyword **condition_compare_value** as well.

Keyword

```
condition_criteria
```

Value

```
BLANK | DEFINED | EQUAL | GREATERTHAN | GREATERTHANOREQUAL |
INGROUP | LESSTHAN | LESSTHANOREQUAL | NONBLANK | NONZERO |
NOTBLANK | NOTDEFINED | NOTEQUAL | NOTGREATERTHAN | NOTINGROUP |
NOTLESSTHAN | NOTZERO | ZERO
```

NON and NOT criteria are equivalent.

DEFINED criteria used to determine if a variable name is known to the web server for this HTTP request.

A variable is DEFINED for:

- HTML Form variables that are returned (empty text inputs are not returned)
- HSM File Upload processing (e.g., NewFile.handle for HTML file input tag "NewFile")
- Query variables included in the URL (e.g., somepath?var1=val1&var2=var2)
- Explicit assignments anywhere in the HSM Resource file (e.g., assign_field_value=var,"val")
- Assignments in **[REPLY]** groups (e.g., start_length_field=*,*,var)

In most cases, there is no difference between a variable being DEFINED with a current value of blank and it being UNDEFINED. This equivalence is actively used in handling blank form fields – these are not normally returned by the browser, and the corresponding HSM variable is therefore UNDEFINED. HSM programmers do not need to do anything special to correctly handle such data. The DEFINED test is available to know if a variable is blank because it was explicitly set so or because it has never been set to anything.

Allowable Groups

```
[DO]   [BLOCK]   [REPLY]   [SERVER REQUEST]   [SUBMIT BUTTON]
```

Examples

(See condition_variable)

condition_compare_value

condition_compare_value is used in conjunction with keywords condition_variable and **condition_criteria** as the value to compare the variable to, as defined by **condition_variable**.

Keyword

```
condition_compare_value
```

Value

```
field-name | number | literal
```

Allowable Groups

```
[DO]   [BLOCK]   [REPLY]   [SERVER REQUEST]   [SUBMIT BUTTON]
```

Examples

(See condition_variable)

export_field

export_field is used to embed a field and its value as hidden inputs into an HTML form immediately preceding the closing </FORM> tag. The name and value of the hidden input field are the same as the name and value of the exported field.

When using **export_field** with arrays, only entire arrays and simple fields can be exported. If array elements must be exported, use the **assign_field_value** keyword to create a simple variable to export in place of the array element.

Keyword

```
export_field
```

Value

```
field-name
```

Allowable Groups

```
[DO]   [REPLY]   [SERVER REQUEST]   [SUBMIT BUTTON]
```

Examples

```
[REPLY]
opcode= GETLIST
start_length_field=   1,    3.0, NumberReturned
export_field= NumberReturned
```

results in NumberReturned being embedded into the HTML as <INPUT NAME=NUMBERRETURNED TYPE=HIDDEN VALUE="20">, if the actual value for NumberReturned is 20.

name

name is used for two different Groups, **[BLOCK]** and **[SUBMIT BUTTON]**.

For the **[BLOCK]** group, it is used to identify the **[BLOCK]** in the HSM file that corresponds to the HSMBLOCK element in the HTML file.

For the **[SUBMIT BUTTON]** group, it is used to identify the Submit Button clicked on an HTML form, <INPUT type="submit" name="ButtonName" value="ButtonValue">.

Maximum length for the block name is 31 characters.

Keyword

```
name
```

Value

```
field-name
```

Allowable Groups

```
[BLOCK]   [SUBMIT BUTTON]
```

Examples

```
[BLOCK]
name= LISTBLOCK
condition_variable= ShowList
condition_criteria= EQUAL
condition_compare_value= "Y"
repeat_count= NumberReturned
```

results in the corresponding HTML section to repeat its contents NumberReturned times provided ShowList is equal to 'Y'. In this example **repeat_count** is conditionally executed based upon the value of ShowList.

```
[SUBMIT BUTTON]
name= UPDATE
assign_field_value= ButtonClicked, "UPDATE"
```

…results in conditionally assigning the variable ButtonClicked the literal value 'UPDATE' when the submit button UPDATE from an HTML form is clicked.

opcode

opcode is used to identify a function or subroutine to execute within a given HSM server. Maximum length for the **opcode** value is 10 characters.

If a variable is used for the Opcode then it must be prefixed with the ampersand (&) character to notify Strategi the value is a variable and not a literal. This ensures backwards compatibility with previous versions of Strategi that do not support variable Server and Opcode names.

On server requests and replies, the Opcode is converted to uppercase before sending to or receiving from the HSM server program.

Keyword

```
opcode
```

Value

```
&field-name | literal
```

Allowable Groups

```
[REPLY]  [SERVER REQUEST]
```

Examples

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETLIST
```

results in an HSM server request to MYSERVER.

```
[DO]
assign_field_value= VAROPCODE, "GETLIST"

[SERVER REQUEST]
server= MYSERVER
opcode= &VAROPCODE
```

results in an HSM server request to MYSERVER with opcode GETLIST.

repeat_count

repeat_count is used to control **[BLOCK]** repeating behavior. Useful when needing to replicate a section of HTML containing array variables populated on a server reply. Typical use is showing lists of items in conjunction with HTML tables (e.g., <TABLE>, <TR>, <TD> tags).

Keyword

```
repeat_count
```

Value

```
field-name | number | literal
```

Allowable Groups

```
[BLOCK]
```

Examples

```
[BLOCK]
name= LISTBLOCK
condition_variable= ShowList
condition_criteria= EQUAL
condition_compare_value= "Y"
repeat_count= NumberReturned
```

results in the corresponding HTML section to repeat its contents NumberReturned times provided ShowList is equal to 'Y'. In this example **repeat_count** is conditionally executed based upon the value of ShowList.

```
[BLOCK]
name= LISTBLOCK
condition_variable= ShowList
condition_criteria= EQUAL
condition_compare_value= "Y"
repeat_count= 20
```

results in the corresponding HTML section to repeat its contents 20 times provided ShowList is equal to 'Y'. In this example **repeat_count** is conditionally executed based upon the value of ShowList.

```
[BLOCK]
name= LISTBLOCK
condition_variable= ShowList
condition_criteria= EQUAL
condition_compare_value= "Y"
repeat_count= "5"
```

results in the corresponding HTML section to repeat its contents 5 times provided ShowList is equal to 'Y'. In this example repeat_count is conditionally executed based upon the value of ShowList.

```
[BLOCK]
name= LISTBLOCK
repeat_count= "5"
```

results in the corresponding HTML section to repeat its contents 5 times. In this example LISTBLOCK is always repeated 5 times and never conditionally executed.

return_http_forbidden

return_http_forbidden causes the HTTP error response of 403, Forbidden, to be sent with a structured HTML page that shows Code and Text in the normal error details. The "Strategi Error" value placed on the HTML page will be the Code value set via this keyword, and the "Error Text" value will be that of Text.

If a variable is used for the Code or Text components, it must be prefixed with the ampersand (&) character to notify Strategi to treat the value as a variable and not a literal.

A common use of this keyword is in HSM server replies where the desired response simulates an HTTP 403 Forbidden error.

Keyword

```
return_http_forbidden
```

Value

```
code, text

  where
    code= &field-name | integer
    text= &field-name | literal
```

Allowable Groups

```
[DO]  [REPLY]  [SERVER REQUEST]  [SUBMIT BUTTON]
```

Examples

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETINFO

[REPLY]
opcode= BADERROR
start_length_field=   1,   10, ErrorCode
start_length_field=   *,   50, ErrorText
return_http_forbidden= &ErrorCode, &ErrorText
```

results in an HTTP error response of 403, but with the ErrorCode and ErrorText specifically set by the HSM server MYSERVER upon a **[REPLY]** Opcode of BADERROR.

return_http_response

return_http_response causes the HTTP response of Code to be sent with a structured HTML page that shows Code and Text in the normal error details. The "Strategi Error" value placed on the HTML page will be the Code value set via this keyword, and the "Error Text" value will be that of Text.

Optionally, a third component of value can contain a URL value for HTTP redirection. The Code must be a valid HTTP rediection code of 301 (Moved Permanently) or 302 (Moved Temporarily).

If a variable is used for the Code or Text components, it must be prefixed with the ampersand (&) character to notify Strategi to treat the value as a variable and not a literal.

Keyword

```
return_http_response
```

Value

```
code, text, URL (optional)

  where
    code= &field-name | integer
    text= &field-name | literal
     URL= literal
```

Allowable Groups

```
[DO]   [REPLY]   [SERVER REQUEST]   [SUBMIT BUTTON]
```

Examples

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETINFO

[REPLY]
opcode= BADERROR
start_length_field=   1,   10, ErrorCode
start_length_field=   *,   50, ErrorText
return_http_forbidden= &ErrorCode, &ErrorText
```

results in an HTTP error response based upon the value of variable ErrorCode with the ErrorText and ErrorCode specifically set by the HSM server MYSERVER upon a **[REPLY]** Opcode of BADERROR.

server

server is used to identify the HSM server for processing a **[SERVER REQUEST]** group.

HSM server names are defined to Strategi using the CRTHSMSVR command. Maximum length for the server name is 8 characters.

If a variable is used for Server, it must be prefixed with the '&' character to notify Strategi to treat the value as a variable and not a literal. This ensures backwards compatibility with previous versions of Strategi that do not support variable Server and Opcode names. The variable must be defined prior to the **[SERVER REQUEST]** or an error will result because Strategi will treat `&field-name` as a literal.

On server requests and replies, the Server name is converted to uppercase before sending to the HSM server program.

Keyword

```
server
```

Value

```
&field-name | literal
```

Allowable Groups

```
[SERVER REQUEST]
```

Examples

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETLIST
```

results in an HSM server request to MYSERVER.

```
[DO]
assign_field_value= VARSVR, "MYSERVER"
```

results in VARSVR having the value MYSERVER so that,

```
[SERVER REQUEST]
server= &VARSVR
opcode= GETLIST
```

results in an HSM server request to MYSERVER.

start_length_field

start_length_field defines the buffer position on HSM server requests and replies. This keyword's value is comprised of three components: StartPosition, Length, and value-to-set-or-get, which can be represented by variable or literal values. On server requests **start_length_field** is used to set a value in the HSM buffer; on server replies **start_length_field** is used to get a value from the HSM buffer.

The first StartPosition is 1. If the StartPosition is the '*' character, Strategi will assume it to be 1 if it is the first **start_length_field** keyword/value pair in the server request or reply. For server requests and replies containing multiple **start_length_field** keyword value pairs, a '*' character for the StartPosition will inform Strategi to dynamically calculate the StartPosition based upon the prior StartPosition plus Length.

Length, if represented by an array, is of the form "number-of-elements X element-size" (e.g., "10x20"). The first array index is 1 (some programming languages refer to the first array index as 0). Length can also be represented in decimal form where it is considered zoned decimal format (e.g., 5.1).

On server requests 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.

On replies the variables named in the "value-to-set-or-get" component of **start_length_field** are populated based upon the StartPosition and Length components. In essence, the **start_length_field** keyword is an instruction to Strategi to get data from the reply buffer beginning with StartPosition through Length and populate the `field-name` defined.

Keyword

```
start_length_field
```

Value

```
StartPosition, Length, field-name | literal

  where
    StartPosition= * | integer
    Length=        array | number
```

Allowable Groups

```
[REPLY]  [SERVER REQUEST]
```

Examples

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETDETAILS
```

```
start_length_field=   *,  15, PartNumber
```

results in populating the first 15 characters of the request buffer with the value of the PartNumber field.

```
[REPLY]
opcode= GETDETAILS
start_length_field=   1,   40, PartDescription
start_length_field=   *,   10, UnitOfMeasure
start_length_field=   *, 11.2, UnitCost
start_length_field=   *, 11.2, UnitPrice
```

results in poplulating PartDescription with the value of the reply buffer from position 1 through 40; UnitOfMeasure with the value of the reply buffer from position 41 through 50; UnitCost with the value of the reply buffer from position 51 through 61; and, UnitPrice with the value of the reply buffer from position 62 through 72.

```
[SERVER REQUEST]
server= MYSERVER
opcode= GETLIST
start_length_field=   1,  15, Category

[REPLY]
opcode= GETLIST
start_length_field=   1,    3.0, NumberReturned
start_length_field=   4,      1, ShowList
start_length_field=   5,  20x15, PartNbrArray
start_length_field= 305,  20x40, PartDscArray
start_length_field=1105,  20x10, UnitOfMeasureArray
start_length_field=1305,20x11.2, UnitCostArray
start_length_field=1525,20x11.2, UnitPriceArray
```

results in poplulating various array fields, PartNbrArray, PartDscArray, etc., where the number of elements per field is defined as 20. The nth element of PartNbrArray is referred to by the following syntax:

```
PartNbrArray__n -or- PartNbrArray[n]
```

which allows array extraction or population via the **assign_field_value** keyword:

```
assign_field_value= PartNbr1, PartNbrArray__1
assign_field_value= PartDsc1, PartDscArray__1
```

where element 1 of PartNbrArray is extracted; and, for array population,

```
assign_field_value= PartNbrArray__18, "999ABC"
```

element 18 of PartNbrArray is populated with the value of 999ABC.

substitute_url

substitute_url causes processing to switch immediately to the specified URL value, which will be an HTML document or HSM file on the same server. If the URL value contains the colon (:) character, or is not an HTML or HSM file, it becomes a normal HTTP redirection.

substitute_url is generally the last line in a group due to the "goto" nature of the keyword, and hence, no further keyword or group processing continues on that page.

Keyword

```
substitute_url
```

Value

```
field-name | literal
```

Allowable Groups

```
[DO]  [REPLY]  [SERVER REQUEST]  [SUBMIT BUTTON]
```

Examples

```
[DO]
substitute_url= list.htm
```

causes group processing to end and immediately request list.htm.

```
[REPLY]
opcode= Success
start_length_field=   1,1024, nextURL
substitute_url= nextURL
```

causes group processing to end and immediately request the URL contained in the variable nextURL.

Note that the Pocket Strategi implementation of HSM requires that a field name be prefixed with an ampersand to avoid confusion between a field-name and a literal. For the example above,

```
[REPLY]
opcode= Success
start_length_field=   1,1024, &nextURL
```
substitute_url= &nextURL

HSM Special Values

HSM Special Values are Strategi-supplied HSM variables available during HSM processing. Since they are variables they can be used in HSM Resource files as well as HTML documents.

HSM Special Value Chart

The following is a list of HSM Special Values:

HSM Special Value	**Description**
*CLIENT.IPADDRESS	The requesting client's IP address
*CLIENT.USERAGENT	The HTTP browser identification
*DATE	Current date, CCYYMMDD
*DATE.CCYY	Current date 4-digit Year, CCYY

*DATE.CENTURY	Current date 2-digit Century, CC
*DATE.DAY	Current date 2-digit Day, DD
*DATE.MONTH	Current date 2-digit Month, MM
*DATE.YEAR	Current date 2-digit Year, YY
*DATETIME	Current date and time, CCYYMMDDHHMMSS
*EVEN_INDEX	HSMBLOCK name that evaluates to "true" for an even index value on an iteration of an HSMBLOCK
*HOST.NAME	Value from RTVSYSVAL SYSNAME, e.g. 'DEV400'
*HOST.OSVERSION	Value of DTAARA QSYS/QSS1MRI, e.g. 'V4R2M0'
*HOST.SERIALNUMBER	Value from RTVSYSVAL QSRLNBR, e.g. '1211221'
*HOST.COUNTRY	Value from RTVSYSVAL QCNTRYID 'US'
*HOST.MODEL	Value from RTVSYSVAL QMODEL, e.g. 'S20' (leading spaces dropped)
*HOST.PROCESSOR	Value from RTVSYSVAL QPRCFEAT, e.g. '2165'
*HTTPHEADER.<httpheader>	Retrieve the corresponding value for the HTTP Header keyword (defined by <httpheader>)
*HTTPHEADERLIST.KEYWORDS	Lists all HTTP Header keywords, each surrounded by '[ ]'
*HTTPHEADERLIST.VALUES	Lists all HTTP Header keyword values, each surrounded by '[ ]'
*HTTPHEADERLIST.KEYWORDS_AND_VALUES	Lists all HTTP Headerkeywords and values, each surrounded by '[ ]'
*INDEX	Index value for an iteration of an HSMBLOCK
*ODD_INDEX	HSMBLOCK name that evaluates to "true" for an odd index value on an iteration of an HSMBLOCK
*REPLY.OPCODE	The current reply Opcode (useful if opcode=*OTHER was specified)
*REQUEST.OPCODE	The Opcode of the last HSM request made
*REQUEST.SERVER	The name of the HSM server to which the last request was made
*STRATEGI.USERS	Strategi value USER#, eg.'000000235'
*STRATEGI.VERSION	Strategi value RELEASE, e.g. '132'
*STRATEGI.VERSION_M	The modification portion of the Strategi value RELEASE
*STRATEGI.VERSION_R	The release portion of the Strategi value RELEASE

*STRATEGI.VERSION_V	The version portion of the Strategi value RELEASE
*TIME	HHMMSS (24hr clock)
*TIME.HOUR12	HH (12hr clock)
*TIME.HOUR12_AMPM	'AM' or 'PM' uppercase English
*TIME.HOUR24	HH (24hr)
*TIME.MINUTES	MM
*TIME.SECONDS	SS
*USER.REFERER	The URL from which this request was made
*WEBSITE.IPADDRESS	The server-side IP address for the request
*WEBSITE.NAME	The name of the website the request was made on
*ZONE.CODE	The zone name for where this page is located
*ZONE.DOMAIN	The name of the Strategi domain (*IFS or website name) where this page is located
*ZONE.NAME	The zone name for where this page is located (alternative to *ZONE.CODE)
*ZONE.TEXT	The descriptive text for this Strategi zone
*ZONE.TYPE	The zone type for where this page is located

The following require the website 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 logged in user
*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*	Where *ua* is the name of the Strategi User Attribute to retrieve

The following two examples demonstrate HSM Special Value usage.

Example: Display a user's TCP/IP address on a web page using *CLIENT.IPADDRESS as part of the raw HTML code.

homepage.htm

```
<HTML>
<BODY>
Client IP Address: <HSM name="*CLIENT.IPADDRESS">
</BODY>
</HTML>
```

homepage.hsm

```
[DO]
```

the client IP address is substituted for the <HSM> tag during HSM processing. HOMEPAGE.HSM exists solely to invoke HSM processing.

Example: Display a user's TCP/IP address on a web page using *CLIENT.IPADDRESS as an HSM variable within an HSM Resource file.

homepage.htm

```
<HTML>
<BODY>
Client IP Address: <HSM name="UsersIPAddress">
</BODY>
</HTML>
```

homepage.hsm

```
[DO]
assign_field_value= UsersIPAddress, *CLIENT.IPADDRESS
```

the client IP address is stored in another variable, UsersIPAddress, and used in the HTML file for display. Using this method allows for additional variable manipulation within the HSM Resource file.

Additional Group and Keyword Examples

The following examples show various ways to implement HSM groups and keywords.

Using DEFINED Criteria

Undefined variables are considered blank, and a separate DEFINED test allows the determination of a defined variable. In the case of Submit Buttons, they are simply variables that have a value when clicked. If they are not clicked, they have no value when the HTML form is submitted for processing, and therefore, an alternative method of determing if a button is clicked is to perform a DEFINED test.

The next two examples are different methods of accomplishing the same result.

Using the **[SUBMIT BUTTON]** group to determine if button clicked:

```
[SUBMIT BUTTON]
name=ButtonXYZ
assign_field_value= SomeVariable, "ButtonXYZ PRESSED"
```

Using the DEFINED test to determine if button clicked:

```
[DO]
condition_variable= ButtonXYZ
condition_criteria= DEFINED
assign_field_value= SomeVariable, "ButtonXYZ Pressed"
```

the variable ButtonXYZ has a value and is defined in the HTML. Since the button has a value associated with it, that value could be used in HSM processing as well (not shown here).

Array Elements

When a variable has been defined as an array, individual elements can be explicitly referred to in the HTML as "FieldName __ n" ('__' is a double-underscore with no blank separator) or "FieldName[n]" for element number "n".

The first element of the array is element 1, and specific array elements can be set or retrieved in the HSM file. Additionally, the array index can be represented by a variable, but must be preceded by the '&' character to notify Strategi to treat it as a variable.

For example, the **[REPLY]** group defined by Opcode GOTPRICING demonstrates setting the variable FirstItem to the value of ItemNumber array's first element.

```
[REPLY]
opcode= GOTPRICING
start_length_field=   1,      3.0, NumberReturned
start_length_field=   4,    10x15, ItemNumber
start_length_field= 154, 10x50.2, ItemPrice
assign_field_value= FirstItem,     ItemNumber[1]
assign_field_value= FirstPrice,    ItemPrice[1]
```

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 "double-underscore" style, the HTML would read as:

```
<INPUT type="text" name="FirstItem__1">
```

and in the "square brackets" style,

```
<INPUT type="text" name="FirstItem[1]">
```

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

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.

The first repetition of the block would replace *INDEX with 1, the second with 2, and so on. For example:

```
<TABLE>
<HSMBLOCK name="ItemLoop">
<TR><TD>
Number <HSM name="*INDEX">: <HSM name="ItemName">
</TD></TR>
</HSMBLOCK>
</TABLE>
```

A very popular use of *INDEX is dynamically generating unique variable names which can be referenced using Javascript. For example:

```
<FORM name="Main" method="post" action="list.htm">
<TABLE>
<HSMBLOCK name="ItemLoop">
<TR><TD>
Item: <HSM name="ItmNumb">
<INPUT type="hidden" name="ItmNumb__<HSM name="*INDEX">"
  value="<HSM name="ItmNumb">">
</TD>
</TR>
</HSMBLOCK>
</TABLE>
</FORM>
```

after the first iteration of the loop:

```
<FORM name="Main" method="post" action="list.htm">
<TABLE>
<TR><TD>
Item: 999ABC
<INPUT type="hidden" name="ItmNumb__1" value="999ABC">
</TD></TR>
</TABLE>
</FORM>
```

the first element of the ItmNumb array is replaced by its specific value (e.g., 999ABC).

after the second iteration of the loop:

```
<FORM name="Main" method="post" action="list.htm">
<TABLE>
<TR><TD>
Item: 999ABC
<INPUT type="hidden" name="ItmNumb__1" value="999ABC">
</TD></TR>
<TR><TD>
Item: 999ABD
<INPUT type="hidden" name="ItmNumb__2" value="999ABD">
</TD></TR>
</TABLE>
</FORM>
```

the first and second elements of the ItmNumb array are replaced by their specific values (e.g., 999ABC and 999ABD).

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 blue and red.

Conditional [DO]

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

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>
```

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.

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 (e.g., a blank space is %20). This is an HTTP requirement and not a restriction imposed by Strategi.

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 a .HSM file as well as an HTML file.

HTTP Header Special Values

HSM allows you to retrieve the HTTP Header information and use it directly on a web page, or pass the information onto an HSM server request for manipulation and extraction by a server program.

Using *HTTPHEADER.REFERER directly on the HTML page:

```
<P>Referring URL: <HSM name="*HTTPHEADER.REFERER">
```

results in the HTML page displaying the URL from the referring page,

```
<P>Referring URL: http://www.someURL.com/somepage.htm
```

Using *HTTPHEADER.REFERER in a server request:

```
[SERVER REQUEST]
Server= MYSERVER
opcode= GETURLINFO
start_length_field=   1,9999, *HTTPHEADER.REFERER
```

results in the value for *HTTPHEADER.REFERER being sent in the server request and available to the HSM server program for manipulation. This specal value is particularly useful if compiling statistics where users came from to get to this web page.

HSM Error Messages

Strategi returns HSM processing errors, such as invalid KEYWORD=VALUE pairs, as HTTP 523 and 524 error messages. HTTP codes 523 and 524 are used to distinguish them from true server errors, such as code 500 (Internal Server Error).

Strategi will place the descriptive error message text on the HTML page enclosed in HTML comments. Perform a "View Source" on the HTML page to see the error message details. Typically errors encountered are for code 523, malformed HSM Resource file structure.

HTTP Code Definitions – Strategi Specific

The following HTTP codes are Strategi-specific and are related to HSM processing:

523 – Malformed HSM Resource File
524 – Invalid Use of HSM and/or HSMBLOCK Tags

HTTP Error Substitution to a URL

To substitute a specific resource on a specific HTTP error from a URL hyperlink, *ERROR_HTTP_nnn can be embedded in the URL of the requesting hyperlink to cause the resource substitution:

```
*ERROR_HTTP_nnn=<url>
```

or alternatively,

```
*ERROR_SGI_nnn=<url>
```

This would be included in the HTML. For example, if 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">Link Text</A>
```

Then, if "part0001.htm" did not exist, instead of an HTTP 404 error occurring the user would be served "part_dft.htm".

Chapter 4. HSM Server Design

Server design is not complicated to sketch out. A basic HSM server written in Control Language (CL) can be as little as a dozen lines of code.

The server program is structured to receive, process and reply to requests from clients. Strategi provides a convenient mechanism for transmitting the requests and replies. Each HSM server will utilize the Strategi Application Programming Interface (API) in order to receive a request and send a reply. The syntax for interacting with the API will depend upon the programming language the HSM server is written in.

The goal is to build a fast, secure, transaction-based application that enables users to access information in a controlled and methodical manner.

A strict request/reply protocol must be observed between client and server. For every client request there must be a resulting reply from the server. This request/reply structure is achieved with "Opcodes" – 10-character length strings used to identify functions or subroutines to execute within a given HSM server program. Strategi's naming requirement is such that Opcodes beginning with an asterisk (*) are reserved and must be used in accordance with pre-defined rules.

Design Objectives

By adhering to these design objectives when developing HSM Servers, you will have built a high-performance, dynamic data messaging tool to web-enable critical functions:

- "Black Box" the Strategi system, and therefore the iSeries, from the outside world. Never require clients to have specific knowledge of underlying file structures, database formats, etc.
- 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".
- Design the server to be "stateless" -- not retaining specific knowledge from request to request. This will allow it to be safely parallel-tasked.

  If the client is creating an order which might be comprised of several transactions, then requires final confirmation of the order, have the server return a "handle" to the client. This might actually be some type of database key value, but to the client, it is just an abstract handle required on subsequent requests. The server can then safely deal with multiple clients and need not rely upon "continuity of conversation" since it will not have any client specific variables stored beyond the life of an individual transaction.
- Rationalize client requirements to a set of structured requests the server, or servers, can reply to individually.

Server Pseudo Code

The basic design and flow of an HSM server can be pseudo coded as the following:

```
Main Line {
    Clear STOP-FLAG
    While STOP-FLAG is not set {
        Clear Request & Reply Elements
        Receive Request
        Test Opcode {
            Opcode = "*PING"       Call Ping Subroutine
            Opcode = "*STOP"       Call Stop Subroutine
            (application opcodes)
            Default                Call Bad Opcode Subroutine
            }
        Send Reply
        }
    }
Ping Subroutine {
    Move request opcode to reply opcode
    Move server identification to reply data
    Move length of server id to reply length
    }

Stop Subroutine {
    if (Client Type is "*CONTROL") {
         Move "*STOPPED" to reply Opcode
         Set STOP-FLAG
         }
    else {
        Call Bad Opcode
        }
    }

Bad Opcode Subroutine {
    Move "*ERROR" to reply Opcode
    Move error detail to reply data
    Set reply length
    }
```

Message Elements and Server Interface Calls

The interface calls necessary to implement an HSM server are structured according to programming language. The interface calls and their corresponding message elements are described in the following sections.

The server message elements used in the interface calls are described according to the type of interaction. There are two types of interactions: receiving a request and sending a reply.

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 ID	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
```

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

ILE RPG/CL Program Bound Set

There are three program sets for ILE RPG/CL programs: receiving requests, sending replies and setting server options.

HSMRCVRQS

HSMRCVRQS is the ILE RPG/CL interface for receiving HSM requests. There are five required parameters for the call: Opcode, Data, Data_Length, Client_Type, and Client_ID. Data_Length, Client_Type and Client_ID are set by Strategi, but the parameter values are available within the program. The optional parameters, User_Attr and Client_Location, are enabled via the HSMSVROPT interface call. The optional parameters, when used, must be placed within the parameter list in the order HSMSVROPT was called to enable them.

```
HSMRCVRQS
    OPCODE          *CHAR   10
    DATA            *CHAR 9999
    DATA_LENGTH     *DEC     4
    CLIENT_TYPE     *CHAR   10
    CLIENT_ID       *CHAR   10

  Optional parameters:
    CLIENT_LOCATION *CHAR  128
    USER_ATTRIBUTE  *CHAR  128
    AUTH_TYPE       *CHAR  128
    AUTH_ID         *CHAR  128
```

HSMSNDRPY

HSMSNDRPY is the ILE RPG/CL interface for sending HSM replies. There are three parameters for the call: Opcode, Data and Data_Length.

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

HSMSVROPT

HSMSVROPT is the ILE RPG/CL interface for enabling various server options. There are three parameters for the call: Keyword, Value and Return. Value is dependent upon the Keyword. Some Keywords do not require an associated Value. Return is set to '1' if option setting succeeds, '0' if it fails. The conditions for option setting failure are dependent upon the option.

```
HSMSVROPT
    KEYWORD          *CHAR   20
    VALUE            *CHAR  500
    RETURN           *CHAR    1
```

C Program Bound Set

There are three server-related functions for C programs: receiving requests, sending replies and setting options.

hsm_rcvrqs

hsm_rcvrqs function for receiving HSM requests. There are five required parameters: opcode, data, data_len, client_type, and client. The optional parameters, client_loc, usratr, aut_type, and aut_id are enabled via the hsm_svropt() call. When the optional parameters are used they are expected in the same order as the the calls to hsm_svropt().

```
hsm_rcvrqs(
    HSM_BYTE          *opcode,       /*   10 */
    void              *data,         /* 9999 */
    decimal(4)        *data_len,
    HSM_BYTE          *client_type,  /*   10 */
    HSM_BYTE          *client_id,    /*   10 */

    /* Optional parameters: */
    HSM_BYTE          *client_loc,   /*  128 */
    HSM_BYTE          *usratr,       /*  128 */
    HSM_BYTE          *aut_type,     /*  128 */
    HSM_BYTE          *aut_id        /*  128 */
    );
```

hsm_sndrpy

hsm_sndrpy is the C function for sending HSM replies. There are three required parameters: opcode, data, data_len, client_type, and client.

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

hsm_svropt

hsm_svropt is the C function for enabling various server options. There are three parameters for the call: optkwd, optval and optrtn. optval is dependent upon optkwd. Some optkwd do not require an associated optval. optrtn is set to '1' if option setting succeeds, '0' if it fails. The conditions for option setting failure are dependent upon the option.

```
hsm_svropt(
    HSM_BYTE          *optkwd,       /*   20 */
    HSM_BYTE          *optval,       /*  500 */
    HSM_BYTE          *optrtn        /*    1 */
    );
```

Java Set

Refer to the online "Java API Documentation" Strategi Destination located on the RESOURCES web site of your Strategi installation.

External Set 1: OPM Programs

HSMRCVRQ

HSMRCVRQ is the OPM interface for receiving HSM requests. There are five required parameters for the call: Opcode, Data, Data_Length, Client_Type, and Client_ID. Data_Length, Client_Type and Client_ID are set by Strategi, but the parameter values are available within the program.

```
HSMRCVRQ
    OPCODE           *CHAR   10
    DATA             *CHAR 9999
    DATA_LENGTH      *DEC     4
    CLIENT_TYPE      *CHAR   10
    CLIENT_ID        *CHAR   10
```

HSMSNDRP

HSMSNDRP is the OPM interface for sending HSM replies. There are three parameters for the call: Opcode, Data and Data_Length.

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

HSMSVROP

HSMSVROP is the OPM interface for enabling various server options. There are three parameters for the call: Keyword, Value and Return. Value is dependent upon the Keyword. Some Keywords do not require an associated Value. Return is set to '1' if option setting succeeds, '0' if it fails. The conditions for option setting failure are dependent upon the option.

```
HSMSVROP
    KEYWORD          *CHAR   20
    VALUE            *CHAR  500
    RETURN           *CHAR    1
```

External Set 2: old-specification OPM server programs (deprecated)

```
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 */
```

Server Interface Call Notes

Bound function calls result in performance improvements over that of calling external programs, so it is recommended HSM server programs be written to take advantage of IBM's ILE environment.

The external APIs are programs and not commands.

Server Option Setting

Setting server-related options that enhance HSM server functionality are performed via calls to hsm_svropt(). The program, or function calls enable optional fields for use within the HSM server program. Call order in the program is important because it determines the order in which fields are made available. This is most apparent for the SETRQSPARM option.

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(). The following server option setting calls are demonstrated using RPG.

A single call to HSMSVROPT setting the User Attribute:

```
CALLB      'HSMSVROPT'
PARM       'SETRQSPARM'          TMPCHR20
PARM       'USRATR'              TMPCHR500
PARM       ' '                   TMPCHR1
```

and the resulting call to HSMRCVRQS becomes:

```
CALLB      'HSMRCVRQS'
PARM                             RQSOPC
PARM                             RQSDTA
PARM                             RQSLEN
PARM                             RQSCTY
PARM                             RQSCID
PARM                             USRATR
```

A call to HSMSVROPT setting the User Attribute, immediately followed by a call to HSMSVROPT setting the Location:

```
CALLB      'HSMSVROPT'
PARM       'SETRQSPARM'          TMPCHR20
PARM       'USRATR'              TMPCHR500
PARM       ' '                   TMPCHR1

CALLB      'HSMSVROPT'
PARM       'SETRQSPARM'          TMPCHR20
PARM       'CLTLOC'              TMPCHR500
PARM       ' '                   TMPCHR1
```

and the resulting call to HSMRCVRQS becomes:

```
CALLB      'HSMRCVRQS'
PARM                          RQSOPC
PARM                          RQSDTA
PARM                          RQSLEN
PARM                          RQSCTY
PARM                          RQSCID
PARM                          USRATR
PARM                          CLTLOC
```

Since the call order to HSMSVROPT determines the parameter list for calls to HSMRCVRQS. Interchanging the prior example with a call to HSMSVROPT setting the Location, immediately followed by a call to HSMSVROPT setting the User Attribute:

```
CALLB      'HSMSVROPT'
PARM       'SETRQSPARM'       TMPCHR20
PARM       'CLTLOC'           TMPCHR500
PARM       ' '                TMPCHR1

CALLB      'HSMSVROPT'
PARM       'SETRQSPARM'       TMPCHR20
PARM       'USRATR'           TMPCHR500
PARM       ' '                TMPCHR1
```

and the resulting call to HSMRCVRQS becomes:

```
CALLB      'HSMRCVRQS'
PARM                          RQSOPC
PARM                          RQSDTA
PARM                          RQSLEN
PARM                          RQSCTY
PARM                          RQSCID
PARM                          CLTLOC
PARM                          USRATR
```

If no call to HSMSVROPT is performed:

the resulting call to HSMRCVRQS becomes:

```
CALLB      'HSMRCVRQS'
PARM                          RQSOPC
PARM                          RQSDTA
PARM                          RQSLEN
PARM                          RQSCTY
PARM                          RQSCID
```

Reserved Opcodes

The following Opcodes are defined by Strategi:

*ECHO

The *ECHO Opcode is provided internally by HSM and is not seen by the server. It allows the "echo-ing" of data on reply from a server request to a valid server. This Opcode is especially useful if needing to breakdown a known formatted data string into smaller variables for manipulation in the HSM or HTML file, and the string is simplistic enough not to require a separate HSM server program for manipulating the string.

HSM Server program code implementation: No
HSM Client program code implementation: No

*ERROR

The*ERROR Opcode is the standard error response and 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 example, "The Opcode is not recognized."

HSM Server program code implementation: Yes
HSM Client program code implementation: Yes

*NOSERVER

Strategi will return this Opcode if the server requested is not currently active. This should not be included in your server program code, since this would never be returned if the server is active.

HSM Server program code implementation: No
HSM Client program code implementation: Yes

*OTHER

Strategi will match any previously not matched reply Opcode when *OTHER is used. *OTHER is an HTML/HSM concept only and not a valid reply Opcode an HSM Server should set. This should be used after all expected [REPLY] group Opcode tests to ensure errors are handled consistently.

HSM Server program code implementation: No
HSM Client program code implementation: Yes

*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. This allows a client to "ping" the server and check that it is connecting to a server it is compatible with.

Strategi does not dictate the length of this text. 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.

HSM Server program code implementation: Yes
HSM Client program code implementation: Yes

*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 if the client type of the request is *CONTROL, 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.

HSM Server program code implementation: Yes
HSM Client program code implementation: No

*TIMEOUT

If an HSM request takes longer than the predetermined timeout period for Strategi in order to send a reply, Strategi will end the request and send as the reply Opcode *TIMEOUT. In essence, the server request has taken too long to perform, and therefore, has timed out. Since Strategi sends the reply of *TIMEOUT, an HSM Server should never attempt to send *TIMEOUT as a reply Opcode.

With advanced HSM Servers, the request timeout length can be set using the HSM Server Setting Option API (e.g., HSMSVROPT).

HSM Server program code implementation: No
HSM Client program code implementation: Yes

User-defined Opcodes

All non-reserved Opcodes are user-defined, predetermined at the time of Server design, and used by the client application. The only restrictions on naming are that the Opcode cannot be longer than ten characters and cannot start with an asterisk because Opcodes beginning with an asterisk are considered reserved for Strategi's specialized usage.

HSM Server Special Values

The following chart details the HSM Server Special Values used within an HSM server program.

Field	**Value**	**Meaning**
Keyword	SETRCVTMO	Specify the Receive Request timeout
	SETRQSPARM	Specify an optional field
	CCSID	Specify CCSID for all HSM elements
	PRPCCSID	Specify CCSID for property elements (e.g., opcodes)
	RQSCCSID	Specify CCSID for request data element
	RPYCCSID	Specify CCSID for reply data elements

Client Type	*CONTROL	Client is the host user profile. Request came from a Strategi command (e.g.,*STOP)
	*STRATEGI	Strategi client. Client ID is the Strategi user number and will be 000000000 for an anonymous user
	*HOSTUSER	iSeries User Profile
Rqs Opcde	*PING	Request server identity
	*STOP	Request server to end
	*TIMEOUT	Receive Request call timed out (no reply)
Rpy Opcode	*ERROR	Error processing opcode
	*PING	Server reply to *PING
	*STOPPED	Server response to *STOP

Server Option		**Value**
SETRCVTMO		A number denoting timeout in seconds
SETRQSPARM		Name of field to be included in Request call: CLTLOC, USRATR, AUTTYP, AUTIDN
CCSID		Number from 0 – 65535, *HEX, *JOB
PRPCCSID		Number from 0 – 65535, *HEX, *JOB
RQSCCSID		Number from 0 – 65535, *HEX, *JOB
RPYCCSID		Number from 0 – 65535, *HEX, *JOB

SETRCVTMO

Server option 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

Server option 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: "USRATR" or "CLTLOC". 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).

This option is only available for the bound set of APIs. OPM programs can not take advantage of this enhanced functionality.

User Attribute (USRATR)

HSM Server configuration determines which Strategi User Attribute to access. When a Strategi user accesses the server, the user-attribute field of the HSM Receive Request API is populated with the contents of the specified user attribute for that user. The field is blank for undefined attributes. By using Strategi User Attributes in this manner, you can automatically associate a Strategi user 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).

Client Location (CLTLOC)

DHSM peer networking allows client requests to originate from a different system from which the HSM Server is running on. The 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.

CCSID Options

Automatic translation of request/reply data is done by HSM. The default CCSID is the CCSID of the job when the request is received or the reply is sent. Data is moved in its CCSID; all property elements are converted to the Strategi internal CCSID (CP1140, US with Euro).

Using server options xxxCCSID with *HEX (binary data) values are not guaranteed supportable between all WebCluster systems. CCSID option setting was first introduced with Strategi V1R8.

Strategi only supports pure Single-Byte Coded Character Set (SBCS) and Double-Byte Coded Character Set (DBCS) code pages. Mixed character set code pages are not permitted. More information regarding Coded Character Set Identifiers (CCSIDs) can be found on IBM's websites (e.g., www.ibm.com). Specifically, the manual on "National Language Support" contains term definitions and tables of CCSIDs.

HSM Server Special Value Notes

All HSM servers are required to support opcodes *PING and *STOP. Strategi uses the *PING opcode to periodically check server status, and the *STOP opcode is used to end the server in a controlled manner. By including *PING and *STOP in all HSM servers written, you provide a consistent method for easily determining server identification and controlled ending of the corresponding job.

Any opcode reply made with an opcode beginning with '*' that is not an HSM Server Special Value will be rejected. Opcodes beginning with '*' are reserved for Strategi, and therefore, must be a Strategi-recognized opcode. It is permissible to use Strategi-defined opcodes in the context for which they are designed.

The required reply data format for *ERROR opcode is the following:

```
  HSM_BYTE  MSGIDN [  7]  /* First 7 positions are message ID number */
  HSM_BYTE  SPACE  [  1]  /* Blank space to separate ID and text     */
  HSM_BYTE  MSGTXT [200]  /* Message text                            */
```

Server Behavior on Error

When an HSM server encounters an error condition it will be terminated. The following are possible critical error conditions:

- 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-standard beginning with '*')

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

Chapter 5. HSM Server Implementation

Prepare the HSM server for registering to Strategi by first creating the server program. Regardless of which programming language the HSM server program is written in, the skeleton source file members located in source file SGIEXAMPLE in the Strategi library are an excellent starting point. They contain examples of Strategi HSM API use and will compile successfully without modification. Generally pre-built with three functional example sub-routines, the example source members reduce the amount of time to become proficient by including general program logic and everything needed to make a successful HSM server.

Assuming this is the first time creating an HSM server, the RPGLE example source member HSMSKLRPG will be used for example purposes. If your source member is different, replace references to HSMSKLRPG with the appropriately named source member.

HSM Server Creation

Creating the HSM server program generally requires two steps: (1) creating the program module (*MODULE), and (2) creating the program object (*PGM). Use the Create HSM Java Server Class (CRTCLSSGI) command for creating Java HSM Servers.

A supporting CL program source member, #MAKE, is located in SGIEXAMPLE and is available for compilation. It provides an alternative method of creating the program object by including the necessary CRTxxxMOD and CRTPGM commands in a CL program that can be easily invoked using a PDM (AS/400 Programming Development Manager) option.

Taking the skeleton RPGLE source member without modifying the contents,

1. Prompt the CRTRPGMOD (Create RPG Module) command
   - Enter the "Module" as HSMSKLRPG in "Library" STRATEGI (assuming Strategi installed into the STRATEGI library)
   - Enter the "Source file" as SGIEXAMPLE in "Library" STRATEGI
   - Additional parameters (F10)
   - Enter the "Debugging views" as *SOURCE (not required but very helpful later on if needing to debug the server)
   - Press Enter
2. Upon successful module creation, prompt the CRTPGM (Create Program) command
   - Enter the "Program" as HSMSKLRPG in "Library" STRATEGI
   - Enter the "Module" as *PGM (not required, but the program will have the same name as the module created in step 1)
   - Additional parameters (F10)
   - Enter the "Binding directory" as SGIBNDDIR in "Library" STRATEGI
   - Enter the "Activation group" as *NEW
   - Press Enter

Once the server *PGM object has been successfully created, the next step is to register the program to Strategi so the HSM server job will get started and begin handling HSM requests.

HSM Server Registration

After successfully creating an HSM server program, whether in CL, RPG or Java, the server program must be registered with Strategi so the STRHSMSVR (Start HSM Server) command starts the associated HSM server batch job in the Strategi subsystem. Parameters associated with the registration process determine whether the server job starts up with the Strategi subsystem start up.

HSM server registration is performed via the CRTHSMSVR (Create HSM Server) command, or by using the appropriate Create command via the Work with HSM Servers screen. The HSM server registration process defines the executable object and job environment to Strategi, so that Strategi can successfully start the associated batch job in the Strategi subsystem.

From the Strategi main menu (SGI),

1. HSM Servers (option 7)
2. Create (F6)
   - Enter a unique "Server Name." This is the name clients will refer to when making HSM requests.
   - Select the "Interface Type." RPG and CL-based servers will use *SRVPGM (*DTAQ is deprecated and will no longer be supported in future releases of Strategi). For example purposes, the following steps refer to registering an RPG-based HSM server. Consult command help text for creating additional Interface Type HSM server programs.
   - Additional parameters (F10)
   - Enter the "Server Program" and "Library" where the "Server Program" is located (use *LIBL only if the library is contained in the Strategi job description, usually named STRATEGI)
   - Enter "Instances." The number of "Instances" determines the number of subsystem jobs for "Server Name" started in parallel in the Strategi subsystem.
   - Select the desired "Autostart" value. Specifies if the HSM server is started upon Strategi subsystem startup.
   - Enter the "Text Description."
   - Enter the "Job Description" and "Library" where the "Job Description" is located (use *LIBL only if the Strategi HSM server job description is to be used, usually named STRATEGIH). Recommend creating and using a separate job description from STRATEGIH to ensure necessary libraries are listed on the "Initial library list" section of the job description definition.
   - Select the desired "Restrict User Access" value. Specifies whether Strategi should check HSM Authority restrictions when processing opcodes for this server.
   - Select the desired "Performance Monitoring" value. Specifies whether performance monitoring will run for this server.
   - Enter a specific "Request Parm User Attribute" value or *NONE. Specifies a Strategi User Attribute whose value is available to the server program via an optional parameter on the server request.

Once the server is registered to Strategi it can be started using the STRHSMSVR command, option 9 from the Work With HSM Servers screen, or automatically upon Strategi restart if the AUTO parameter is set to *YES on the Server definition.

HSM Authority

An HSM server secured using HSM authorities defines restrictions and permissions for Strategi Users, Strategi Groups, or even specific iSeries users (through the *HOSTUSER specification of the Strategi User), with regards to whether they can use certain Opcodes. From the “HSM Servers” (WRKHSMSVR) menu option, or directly using the WRKHSMAUT (Work With HSM Authorities) command, 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 of the CRTHSMSVR or CHGHSMSVR commands. In order for authority checking to have user information to check against, the HSM application’s web pages must reside in an authenticated zone.

See the xxxHSMAUT command set for available options.

HSM Performance Monitoring

HSM Servers can be monitored for performance information. A server's performance is monitored only if 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.

See the xxxHSMPFR command set for available options.

HSM Server Debugging

Strategi provides an option for running your batch server program interactively for debugging purposes. The Start HSM Server (STRHSMSVR) command contains the parameter for running the server interactively.

Debugging an HSM server requires a client application to make an HSM request. This is generally, and most easily, done via an HTTP request from a browser. By creating and using test HTML and HSM pages to support server requests and replies, a test environment set up to invoke an HSM server undergoing debugging will facilitate the process.

HSM Server Debug Example

For example purposes, the following steps refer to debugging an ILE RPG-based HSM server. This is one method for accomplishing the debugging task.

From a command line,

1. Execute the CRTRPGMOD (Create RPG Module) command with "Debugging views" set to *SOURCE
2. Execute the CRTPGM (Create Program) command; ensure SGIBNDDIR is included on the BNDDIR parameter
3. Load the Strategi main menu, SGI (command entry example: GO STRATEGI/SGI)
4. Prompt the STRDBG (Start Debug) command and specify program details
   - Enter *YES for the "Update production files" parameter
   - Press Enter. The HSM server source code should display. Scroll down and add a break point at the HSM Receive Request API.
   - Resume (F12)

5. Prompt the Strategi command STRHSMSVR
   - Enter the "Server Name"
   - Select "Additional parameters" (F10)
   - Enter *YES for the "Run interactively? (for debug)" parameter
   - Press Enter and verify the Display Module Source screen is displayed with the cursor positioned at or near the break point. If not, return to step 1.
6. Make an HSM request that invokes the server, usually done via an HTTP request from a browser
7. Step (F10) through the module source
8. When debugging is complete, select "End program" (F3). If unable to End program because input is inhibited, perform a System Request, option 2, to "End previous request"
9. Execute the ENDDBG (End Debug Mode) command

Once debugging is complete, the server will not be active and must be re-started to become active again. Choose either the STRHSMSVR command option, or select the Start option from the Work with HSM Servers screen.

Chapter 6. Dynamic Web Site Content Implementation

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 the HSM Resource file, obtains the fields to be substituted in the template. This file is a text file and is named "<filename>.HSM", where <filename> is the same as the name of the document template intended to be processed. It must exist in the same directory as the template document, and its presence indicates to the Strategi server that the document is not a static one, 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.

HTML with HSM Reference

The following sections pertain specifically to using HTML as the client interface, but could be extrapolated to include other document types. The primary concept related to HSM is variable substitution.

HTML Elements

There are two HTML elements specific to Strategi's HSM architecture, namely, HSM and HSMBLOCK. It is through the normal use of HTML tags that allow for dynamic (variable) web page data. The HSM and HSMBLOCK elements conform to HTML standards, which allows for normal web page development without the designer seeing HTML error messages.

HSM Element

Start Tag:	Required
End Tag:	Optional

This element is used to define a field in the HTML template that is substituted by the field's actual value prior to the page being served to the requesting browser. The optional end </HSM> tag permits sample text to be inserted at template design time. Any data between the start and end tags will not be included in the final generated HTML. Sample text unrelated to field substitution can be marked with a start/end tag pair, containing sample text inserted between them, with the `name` attribute omitted.

Nesting of <HSM> tags is strictly prohibited. If another start tag is found prior to its end tag, then it is deemed there is no corresponding end tag for the first start tag. Every end </HSM> tag must have a corresponding start <HSM> tag.

Attributes

name
```
name=cdata
This attribute is used to give the HSM element a name that is used
as the field identifier for dynamic text substitution. Maximum name
length is 31 characters.
```

escape
```
escape=HTML | URL | BACKSLASHX | BACKSLASHX25 | BSX | BSX25 | NO |
       NONE
This attribute specifies the type of data escaping to perform on the
variable/value substitution. Its default value is "HTML". If omitted,
HTML escaping will occur.
```

HTML	Normal HTML escaping (default) - "&lt" for "<"
URL	URL escaping - "%3C" for "<"; %20 for " "
BACKSLASHX	Substitution into JavaScript literals. All but alphanumeric escaped – "\x3C"for "<"
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. Some browsers wrongly do a URL-style un-escape 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

Examples
The following examples show various ways to use the opening and closing <HSM> tags to assist at HTML design time. By using the optional closing </HSM> tag, your browser should properly render the text between the opening and closing tags because it is correctly formed HTML. Once the page is loaded into a Strategi-based website and HSM processing is invoked, the text between the tags will be removed by Strategi.

Sample Text: Field Associated
Sample text prior to HSM processing:
```
...
<TR>
<TD>Part Number:</TD>
<TD><HSM name="PartNumber">999AAA</HSM></TD>
```

```
  </TR>
  ...
```

after HSM processing:

```
  ...
  <TR>
  <TD>Part Number:</TD>
  <TD>727ABC</TD>
  </TR>
  ...
```

Sample Text: No Field Associated
Sample text prior to HSM processing:

```
  ...
  <H1>Quote of the Day</H1>
  <HSM>Here is a comment I do not want shown on the resolved HTML
    document</HSM>
  <P>Today is the first day of the rest of my life
  <H2>Quote from Yesterday</H2>
  <P>Do not worry what tomorrow will bring ...
  ...
```

after HSM processing:

```
  ...
  <H1>Quote of the Day</H1>
  <P>Today is the first day of the rest of my life
  <H2>Quote from Yesterday</H2>
  <P>Do not worry what tomorrow will bring ...
  ...
```

Data Escaping: URL
HTML formulation prior to HSM processing:

```
  ...
  <A href="invdetails.htm?PartNumber=<HSM name="PartNumber"
    escape="URL">">
  Click here for part number: <HSM name="PartNumber">999AAA</HSM>
  </A>
  ...
```

after HSM processing:

```
  ...
  <A href="invdetails.htm?PartNumber=135%20ABC">
  Click here for part number: 135 ABC
  </A>
  ...
```

HSMBLOCK Element

Start Tag: Required
End Tag: Required

This element is used to define blocks of HTML statements that can be manipulated by name in the HSM Resource file. Unlike the HSM element, HSMBLOCK elements do not perform variable/field substitution on their own. A couple of common uses are conditionally excluding HTML statements and repeating lines of generated statements (repeating blocks). Since the combination of the HSMBLOCK start <HSMBLOCK> and end </HSMBLOCK> tags define HTML statements to affect, the end </HSMBLOCK> tag is required.

The HSMBLOCK **name** is defined in an associated HSM file(s). If for some reason it has not been defined, the start tag and its matching end tag are ignored. This results in the enclosed HTML to be displayed only once, and exactly as specified in the template HTML when the page is delivered to the browser.

There are limits to the size of blocks and sample text. An HSM repeat block (one that contains the **repeat_count** keyword), including both start and end tags, is limited to 7,000 characters of original HTML. Other HSM blocks and HSM sample text are limited to only 1,000 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.

An extremely useful piece of functionality that occurs with repeat blocks is variable indexing. When HSM variables are used in conjunction with HSMBLOCK, they are automatically indexed. If the indexed element is undefined (i.e., the variables are not defined as an array variables), the variables revert to the simple variable name. This is used extensively in situations where array data is returned from a server request.

As page flow complexity increases, it is important to note restrictions.

- Conditional blocks (ones that contain the Condition_ keywords) can not overlap, however, they can be nested.
- Repeat blocks, whose behavior is significantly more dynamic, cannot be nested.

An example that takes into account these restrictions is a conditional block containing a repeat block containing further conditional blocks, with the outer condition preventing table headers for an empty table and the inner condition preventing empty table rows.

Attributes

name
```
name=cdata
This attribute is used to give the HSMBLOCK element a name that is
used as the block identifier for HTML statement manipulation. Maximum
name length is 31 characters.
```

Examples

The following examples show various ways to use the opening and closing <HSMBLOCK> tags in the HTML. Once the page is loaded into a Strategi-based website and HSM processing is invoked, the text between the tags will remain, be removed, or replicated by Strategi depending upon the type of block conditioning.

Block Conditioning: Evaluates to "True"
Sample text prior to HSM processing:
```
...
<P>If an error occurred, a message will be displayed below</P>
<HSMBLOCK name="ErrorMessage">
<P>An error occurred - please check back later</P>
</HSMBLOCK>
...
```

after HSM processing for a block evaluating to a true condition:

```
...
<P>If an error occurred, a message will be displayed below</P>
<P>An error occurred - please check back later</P>
...
```

Block Conditioning: Evaluates to "False"
Sample text prior to HSM processing:

```
...
<P>If an error occurred, a message will be displayed below</P>
<HSMBLOCK name="ErrorMessage">
<P>An error occurred - please check back later</P>
</HSMBLOCK>
...
```

after HSM processing for a block evaluating to a false condition:

```
...
<P>If an error occurred, a message will be displayed below</P>
...
```

Block Conditioning: Repeat Count
Sample text prior to HSM processing:

```
...
<TABLE>
<HSMBLOCK name="ListItems">
<TR>
<TD>Part Number:</TD>
<TD><HSM name="PartNumberArray">999AAA</HSM></TD>
</TR>
</HSMBLOCK>
</TABLE>
...
```

after HSM processing for a repeat block with a **repeat_count** of 2:

```
...
<TABLE>
<TR>
<TD>Part Number:</TD>
<TD>111AAA</TD>
</TR>
<TR>
<TD>Part Number:</TD>
<TD>111AAB</TD>
</TR>
</TABLE>
...
```

Server Side Includes

Server Side Includes are a powerful way to save time when writing HTML. A Server Side Include is one tag added to an HTML file that points to another file. The target file need not be an HTML file. It could be another HTML file, one that contains header or footer-type information to be included on all documents, or, it could be a TXT file with contents appropriate for the location of the document where the include directive is. The contents of this second file are added to the first HTML file in place of the comment containing the include directive.

For example:

homepage.hsm

```
[DO]
```

homepage.htm

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

header.htm

```
<HTML>
<BODY bgcolor="blue">
```

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

The format of the include directive is as follows:

```
<!--#include file="FilenamePlusOptionalURLData" -->
```

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

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

```
<!--#include virtual="FilenamePlusOptionalURLData" -->
```

There can also be HSM substitution within the #include directive:

```
<!--#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 directly 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 **substitute_url** as normal.

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

SHTM and SHTML 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.

Another useful method for using SHTML files is if HSM processing is required for HSM Special Values that need to be displayed and an HSM file is not required. An example of this is displaying the host model number or current date.

For example:

SHTML: HSM Special Value Use
homepage.shtm

Prior to HSM processing:

```
...
<H1>System Information</H1>
Model Number: <HSM name="*HOST.MODEL">Model Number</HSM>
...
```

after HSM processing:

```
...
<H1>System Information</H1>
Model Number: S20
...
```

SHTML: Variable Passing
In this example a variable/value pair will be passed to an HTML page that must display the variable/value contents without requiring separate HSM processing via an HSM Resource file.

homepage.htm

```
...
<A href="displaydetails.shtm?CARPART=Wheel>
Click here to display details
</A>
...
```

Prior to HSM processing:
displaydetails.shtm

```
...
<H1>Automobile Details Section</H1>
Part: <HSM name="CARPART">Sample Text</HSM>
...
```

after HSM processing:
displaydetails.shtm

```
...
<H1>Automobile Details Section</H1>
Part: Wheel
...
```

Visitor Counter Example

The following "Visitor Counter" example demonstrates (1) displaying a single field containing dynamic data from an AS/400 database file onto an HTM page, and (2) invoking HSM processing for the requested HTM page. There is one primary HTM page and one same-named HSM page.

The field in this example represents a counter variable that gets incremented each time the homepage is displayed.

homepage.htm

```
<HTML>
<TITLE>Hot Steel Products Inc. Home Page</TITLE>
<BODY>
<H1>Hot Steel Products Inc.</H1>
<P>Welcome to our Web Site</P>
You are visitor number <HSM name="VisitorNumber">
</BODY>
</HTML>
```

homepage.hsm

```
* Invoke server to get next visitor number

[SERVER REQUEST]
Server= WEBTOOLS
opcode= COUNTER

* Evaluate replies

[REPLY]
opcode= COUNTER
start_length_field=   1,   9, VisitorNumber

[REPLY]
opcode= *NOSERVER
substitute_url= noserver.htm

[REPLY]
opcode= *OTHER
start_length_field=   1,   *, UnexpectedData
assign_field_value= UnexpectedOpcode, *REPLY.OPCODE
substitute_url= unexpect.htm
```

The HTML template is standard HTML with the exception of the special HSM tag. The HSM tag names a field, VisitorNumber, and stipulates to Strategi where on the HTML page to place the dynamic data via substitution.

The HSM Resource file stipulates which Server (WEBTOOLS) to invoke and Opcode (COUNTER) to pass to that server. It also defines any data required to fully qualify the request.

Strategi makes the server request to WEBTOOLS and maps the reply data to the VisitorNumber field upon a matching reply Opcode of COUNTER. The VisitorNumber field is defined as a 9-byte reply field.

The HTML file requested by the browser will be returned immediately following processing of the HSM Resource file, unless a **substitute_url** KEYWORD=VALUE pair is encountered, in which case the specified URL will be returned instead.

If the server is not active when the server request is made, Strategi will return a reply Opcode of *NOSERVER for ease of error handling. It is recommended to include a test for reply Opcode *NOSERVER following all server requests.

If no matching reply Opcode is encountered, reply Opcode *OTHER is used to trap unexpected or unknown reply Opcodes. Ideally, an unknown Opcode is never encountered; however, structuring the HSM Resource file to catch the unknown Opcode and handle accordingly will prevent an un-desired user interface experience. It is recommended to include a test for reply Opcode *OTHER following all server requests as the last **[REPLY]** group.

Guest Book Example

The following "Guest Book" example demonstrates (1) submitting HTML form variables for HSM processing, and (2) invoking HSM processing without requiring a same-named HTM page. There is one primary HTM page and one differently-named HSM page.

The application is designed so users can enter their name and email address, then click a button on the HTML page to "sign" the guest book. All data is written to the iSeries database by the WEBTOOLS server.

guestbook.htm

```
<HTML>
<TITLE>Hot Steel Products Inc.</TITLE>
<BODY>
<H1>Guest Book</H1>
<FORM method="post" action="addentry.hsm">
Your Name:<BR>
<INPUT name="GuestName" type="text" size="40"><BR>
Email Address:<BR>
<INPUT type="text"   name="EmailAddress" size="40"><BR>
<INPUT type="submit" name="SignIt" value="Sign It">
</FORM>
</BODY>
</HTML>
```

addentry.hsm

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

[SERVER REQUEST]
Server= WEBTOOLS
opcode= ADDGUEST
start_length_field=   1,   40, GuestName
```

```
start_length_field=  41,   40, EmailAddress

* Evaluate replies

[REPLY]
opcode= ADDED
substitute_url= added_ok.htm

[REPLY]
opcode= *NOSERVER
substitute_url= noserver.htm

[REPLY]
opcode= *OTHER
start_length_field=   1,    *, UnexpectedData
assign_field_value= UnexpectedOpcode, *REPLY.OPCODE
substitute_url= unexpect.htm
```

The filename of the HSM Resource file is ADDENTRY.HSM, not GUESTBOOK.HSM. This is correct, since GUESTBOOK.HTM is not a template, and therefore, does not require any HSM processing prior to displaying.

HSM resources are only required to process the form, so ADDENTRY.HSM is specified as the target (i.e., action=…) of the form submission. Strategi does not require ADDENTRY.HTM to exist since only HSM resources (and not the display of an HTML document) are required. However, because the final document rendered must not be an HSM file, a **substitute_url** must be performed to prevent an error from occurring. The document returned to the browser by Strategi is specified by the **substitute_url** KEYWORD=VALUE pair. If one was not specified, or no reply Opcode matched one specified, an error would be displayed. It is recommended to include a test for reply Opcode *OTHER following all server requests as the last **[REPLY]** group.

Stock Inquiry Example

The following "Stock Inquiry" example demonstrates (1) displaying an array of fields containing dynamic data from an AS/400 database file onto an HTM page , (2) invoking HSM processing for the requested HTM page, (3) submitting HTML form variables for HSM processing, (4) invoking HSM processing without requiring a same-named HTM page, and (5) populating variables based upon a specific Submit Button being clicked. There is one primary HTM page and two HSM pages.

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 file is invoked if the form is "submitted" (i.e., if a submit button for the form is clicked).

stock.htm

```
<HTML>
<BODY>
<H1>Stock Inquiry</H1>
<FORM method="post" action="stockFormAction.hsm">
<SELECT name="SelectedItem" size="10">
<HSMBLOCK name="ItemListBlock">
```

```
  <OPTION><HSM name="Items">ExampleData1
  <OPTION>ExampleData2
  <OPTION>ExampleData3
  <OPTION>ExampleData4
  <OPTION>ExampleData5
  <OPTION>ExampleData6
  <OPTION>ExampleData7
  <OPTION>ExampleData8
  </HSM>
</HSMBLOCK>
</SELECT>
<BR>
<INPUT type="submit" name="Prev10"     value="Previous 10">
<INPUT type="submit" name="Next10"     value="Next 10">
<BR><BR>
<INPUT type="submit" name="GetDetails" value="Item Details">
<BR><BR>
<HSMBLOCK name="DetailsBlock">
Stock available is <HSM name="OnHand">44</HSM><BR>
Stock on Backorder is <HSM name="BackOrdered">44</HSM><BR><BR>
</HSMBLOCK>
<INPUT type="submit" name="order" value="Place an Order">
</FORM>
</BODY>
</HTML>
```

stock.hsm

```
* Invoke server to get info for stock

[SERVER REQUEST]
Server= STOCK
opcode= GETDATA
start_length_field=   1,   1, OP
start_length_field=   2,  10, PositioningKey
export_field= ListCount
export_field= Items

* Evaluate replies

[REPLY]
opcode= ITEMLIST
start_length_field=   1,    4, ListCount
start_length_field=   5,10x20, Items

[REPLY]
opcode= ITEMDATA
start_length_field=   1,  20, DetailItem
start_length_field=  21, 5.0, OnHand
start_length_field=  26, 5.0, BackOrdered

* Block Handling

[BLOCK]
name= ItemListBlock
condition_variable= ListCount
condition_criteria= NONZERO
```

```
repeat_count= ListCount

[BLOCK]
name= DetailsBlock
condition_variable= DetailItem
condition_criteria= NONBLANK
```

stockFormAction.hsm

```
* Process Stock form

[SUBMIT BUTTON]
name= Prev10
assign_field_value= OP, "B"
assign_field_value= PositioningKey, Items__1
substitute_url= stock.htm

[SUBMIT BUTTON]
name= Next10
assign_field_value= OP, "F"
assign_field_value= PositioningKey, Items__10
substitute_url= stock.htm

[SUBMIT BUTTON]
name= GetDetails
assign_field_value= OP, "I"
assign_field_value= PositioningKey, SelectedItem
substitute_url= stock.htm

[SUBMIT BUTTON]
name= Order
export_field= SelectedItem
substitute_url= order.htm
```

HSM resources are required to successfully build STOCK.HTM since it is an HTML page containing HSM and HSMBLOCK tags. Prior to building STOCK.HTM, Strategi locates and generates resources according to the directives on STOCK.HSM. The HTML template STOCK.HTM is standard HTML with the exception of the special HSM and HSMBLOCK tags.

The HSM tag names an array, Items, and stipulates to Strategi where on the HTML page to place the dynamic data via substitution. Strategi will dynamically substitute array elements onto the HTML template when used in conjunction with repeating HSMBLOCKs.

The first HSMBLOCK tag, ItemListBlock, defines a block of HTML statements that will be repeated according to the variable ListCount. This type of HSMBLOCK is known as a repeating block since the HTML statements enclosed within the start and end HSMBLOCK tags is replicated according to the **repeat_count** KEYWORD=VALUE pair.

The second HSMBLOCK tag, DetailsBlock, defines a block of HTML statements to conditionally display based upon the variable DetailItem being NONBLANK. This type of HSMBLOCK is known as a conditional block since the HTML statements enclosed within the start and end HSMBLOCK tags is conditionally displayed according to the Condition set of KEYWORD=VALUE pairs.

The HSM Resource file STOCK.HSM stipulates which server (STOCK) to invoke and Opcode (GETDATA) to pass to that server. It also defines the data required to fully qualify the request. Strategi makes the server request to STOCK, mapping the request data to the Op and PositioningKey fields, 1- and 10-byte request fields, respectively. Strategi maps the reply data to the ListCount and Items fields upon a matching reply Opcode of ITEMLIST. Upon a matching reply Opcode of ITEMDATA, Strategi maps the reply data to the DetailItem, OnHand, and BackOrdered fields. The ListCount field is defined as a 4-byte reply field; Items is defined as a 10-element reply array field where each element is 20 bytes; DetailItem is defined as a 20-byte reply field; and OnHand and BackOrdered are both defined as 5-byte numeric reply fields with zero decimals.

As the action attribute of the FORM tag, STOCKFORMACTION.HSM in invoked when one of the four submit buttons are clicked on STOCK.HTM. Because each INPUT tag has a name attribute, and submit buttons are defined as an INPUT tag whose type attribute is SUBMIT, Strategi will determine which submit button was clicked and allow submit button testing in the HSM Resource file, STOCKFORMACTION.HSM. Based upon the submit button clicked, a particular **[SUBMIT BUTTON]** group is processed and the remaining three **[SUBMIT BUTTON]** groups are excluded. This controls field values and page flow based upon the **assign_field_value** and **substitute_url** KEYWORD=VALUE pairs.

Chapter 7. HSM Client Design

Client design is not complicated to sketch out. A basic HSM client written in Control Language (CL) can be as little as a dozen lines of code.

The client program is structured to send, receive and process requests to servers. Strategi provides a convenient mechanism for transmitting the requests and replies. Each HSM client will utilize the Strategi Application Programming Interface (API) in order to send a request and receive a reply. The syntax for interacting with the API will depend upon the programming language the HSM client is written in.

The goal is to build a fast, secure, transaction-based application that enables client applications to access information in a controlled and methodical manner. One frequent use of the client APIs is having one HSM server make an HSM request of another one. The first HSM server becomes a "client" to the second one.

The HSM client can be a stand-alone client application or it can be another HSM server. Regardless of how the HSM client APIs are used, once proficient at writing HSM server and client applications, building complex HSM applications will become much easier.

A strict request/reply protocol must be observed between client and server. For every client request there must be a resulting reply from the server. This request/reply structure is achieved with "Opcodes" – 10-character length strings used to identify functions or subroutines to execute within a given HSM server program. Strategi's naming requirement is such that Opcodes beginning with an asterisk (*) are reserved and must be used in accordance with pre-defined rules.

Design Objectives

Client Pseudo Code

The basic design and flow of an HSM client can be pseudo coded similarly to that of an HSM server program. The details prior to sending the request and how to handle the reply depend upon application functionality. An HSM client functions differently than an HSM server does because of the client/server relationship. A client makes a request to a server and waits for a response; whereas a server is waiting for a client to make a request, then responds.

Client pseudo code example:

```
. . .
Send Request
Receive Reply
Test Opcode {
    Opcode = "*NOSERVER"   Call Error Subroutine
    Opcode = "*ERROR"      Call Error Subroutine
    Opcode = "*TIMEOUT"    Call Timeout Subroutine
```

```
        (application opcodes)
        Default               Call Bad Opcode Subroutine
        }
    . . .
```

Message Elements and Client Interface Calls

The interface calls necessary to implement an HSM client are structured according to programming language. The interface calls and their corresponding message elements are described in the following sections.

The client message elements used in the interface calls are described according to the type of interaction. There are two types of interactions: sending a request and receiving a reply.

Sending a Request

```
      Server                10     The server for the request
      Opcode                10     The request operation code
      Data                  9999   The request data
      Data Length           4.0    The request data length
```

Receiving a Reply

```
      Opcode                10     The reply operation code
      Data                  9999   The reply data
      Data Length           4.0    The reply length
      Timeout               4.0    The number of seconds to wait for
                      the reply
```

There are two bound, one external and one command client interface sets.

ILE RPG/CL Program Bound Set

There are three program sets for ILE RPG/CL programs: sending requests, receiving replies and setting client options.

HSMSNDRQS

HSMSNDRQS is the ILE RPG/CL interface for sending HSM requests. There are four parameters for the call: Server, Opcode, Data, and Data_Length. Server is required to notify Strategi which HSM server to pass the requesting data to.

```
  HSMSNDRQS
      SERVER          *CHAR   10
      OPCODE          *CHAR   10
      DATA            *CHAR 9999
      DATA_LENGTH     *DEC     4
```

HSMRCVRPY

HSMRCVRPY is the ILE RPG/CL interface for receiving HSM replies. There are four parameters for the call: Opcode, Data, Data_Length, and Timeout. The Timeout parameter allows the requesting server to determine the length of time before Strategi should timeout the request.

```
  HSMRCVRPY
```

```
        OPCODE          *CHAR   10
        DATA            *CHAR 9999
        DATA_LENGTH     *DEC     4
        TIMEOUT         *DEC     4
```

HSMCLTOPT

HSMCLTOPT is the ILE RPG/CL interface for enabling various client options. There are three parameters for the call: Keyword, Value and Return. Value is dependent upon the Keyword. Some Keywords do not require an associated Value. Return is set to '1' if option setting succeeds, '0' if it fails. The conditions for option setting failure are dependent upon the option.

```
    HSMCLTOPT
        KEYWORD          *CHAR   20
        VALUE            *CHAR  500
        RETURN           *CHAR    1
```

C Program Bound Set

There are three client-related function sets for C programs: sending requests, receiving replies and setting client options.

hsm_sndrqs

hsm_sndrqs is the C function for sending HSM requests. There are four required parameters: server, opcode, data, and data_len.

```
    hsm_sndrqs(
        HSM_BYTE            *server,       /*   10 */
        HSM_BYTE            *opcode,       /*   10 */
        void                *data,         /* 9999 */
        decimal(4)           data_len
        );
```

hsm_rcvrpy

hsm_rcvrpy is the C function for receiving HSM replies. There are four required parameters: opcode, data, data_len, and timeout.

```
    hsm_rcvrpy(
        HSM_BYTE       *opcode,       /*   10 */
        void           *data,         /* 9999 */
        decimal(4)     *data_len,
        decimal(4)      timeout
        );
```

hsm_cltopt

hsm_cltopt is the C function for setting client options. There are three parameters for the call: optkwd, optval and optrtn. optval is dependent upon optkwd. Some optkwd do not require an associated optval. optrtn is set to '1' if option setting succeeds, '0' if it fails. The conditions for option setting failure are dependent upon the option.

```
    hsm_cltopt(
        HSM_BYTE       *optkwd,       /*   20 */
        HSM_BYTE       *optval,       /*  500 */
        HSM_BYTE       *optrtn        /*    1 */
        );
```

Java Set

Refer to the online "Java API Documentation" Strategi Destination located on the RESOURCES web site of your Strategi installation.

External Set: OPM Programs

There are three program sets for OPM programs: sending requests, receiving replies and setting client options.

HSMSNDRQ

HSMSNDRQ is the OPM interface for sending HSM requests. There are four parameters for the call: Server, Opcode, Data, and Data_Length. Server is required to notify Strategi which HSM server to pass the requesting data to.

```
HSMSNDRQ
    SERVER          *CHAR   10
    OPCODE          *CHAR   10
    DATA            *CHAR 9999
    DATA_LENGTH     *DEC     4
```

HSMRCVRP

HSMRCVRP is the OPM interface for receiving HSM replies. There are four parameters for the call: Opcode, Data, Data_Length, and Timeout. The Timeout parameter allows the requesting server to determine the length of time before Strategi should timeout the request.

```
HSMRCVRP
    OPCODE          *CHAR   10
    DATA            *CHAR 9999
    DATA_LENGTH     *DEC     4
    TIMEOUT         *DEC     4
```

HSMCLTOP

HSMCLTOP is the OPM interface for enabling various client options. There are three parameters for the call: Keyword, Value and Return. Value is dependent upon the Keyword. Some Keywords do not require an associated Value. Return is set to '1' if option setting succeeds, '0' if it fails. The conditions for option setting failure are dependent upon the option.

```
HSMCLTOP
    KEYWORD         *CHAR   20
    VALUE           *CHAR  500
    RETURN          *CHAR    1
```

Command Set: Simple CLP Programs

There is one command set for simple CLP programs: sending requests, receiving replies and setting client options. The command parameter maximum length is 5000 bytes.

SNDRCVHSM

SNDRCVHSM is the command for sending HSM requests and receiving HSM replies. There are seven required parameters: Server, Rqs_Opcode, Rqs_Data, Rqs_Data_Len, Rpy_Opcode, Rpy_Data, Rpy_Data_Len. There are three optional parameters: Rqs_Timeout, Rqs_Data_Ex, Rpy_Data_Ex.

```
SNDRCVHSM
```

```
SERVER          *CHAR   10
RQS_OPCODE      *CHAR   10
RQS_DATA        *CHAR 5000
RQS_DATA_LEN    *DEC     4
RPY_OPCODE      *CHAR   10
RPY_DATA        *CHAR 5000
RPY_DATA_LEN    *DEC     4
-- F10 --
RQS_TIMEOUT     *DEC      4  DFT(60)
RQS_DATA_EX     *CHAR 4999
RPY_DATA_EX     *CHAR 4999
```

Client Interface Call Notes

Bound function calls result in performance improvements over that of calling external programs, so it is recommended HSM client programs be written to take advantage of IBM's ILE environment.

The external APIs are programs and not commands.

Client Option Setting

Setting client-related options that enhance HSM client functionality are performed via calls to hsm_cltopt().

The following client option setting calls are demonstrated using RPG.

A call to HSMCLTOPT setting the CCSID:

```
CALLB      'HSMCLTOPT'
PARM       'CCSID'               TMPCHR20
PARM       '37'                  TMPCHR500
PARM       ' '                   TMPCHR1
```

A call to HSMCLTOPT setting the PRPCCSID:

```
CALLB      'HSMCLTOPT'
PARM       'PRPCCSID'            TMPCHR20
PARM       '*JOB'                TMPCHR500
PARM       ' '                   TMPCHR1
```

Reserved Opcodes

The following Opcodes are defined by Strategi.

*ECHO

The *ECHO Opcode is provided internally by HSM and is not seen by the server. It allows the "echo-ing" of data on reply from a server request to a valid server. This Opcode is especially useful if needing to breakdown a known formatted data string into smaller variables for manipulation in the HSM or HTML file, and the string is simplistic enough not to require a separate HSM server program for manipulating the string.

HSM Server program code implementation: No
HSM Client program code implementation: No

*ERROR

The*ERROR Opcode is the standard error response and 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 example, "The Opcode is not recognized."

HSM Server program code implementation: Yes
HSM Client program code implementation: Yes

*NOSERVER

Strategi will return this Opcode if the server requested is not currently active. This should not be included in your server program code, since this would never be returned if the server is active.

HSM Server program code implementation: No
HSM Client program code implementation: Yes

*OTHER

Strategi will match any previously not matched reply Opcode when *OTHER is used. *OTHER is an HTML/HSM concept only and not a valid reply Opcode an HSM Server should set. This should be used after all expected [REPLY] group Opcode tests to ensure errors are handled consistently.

HSM Server program code implementation: No
HSM Client program code implementation: Yes

*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. This allows a client to "ping" the server and check that it is connecting to a server it is compatible with.

Strategi does not dictate the length of this text. 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.

HSM Server program code implementation: Yes
HSM Client program code implementation: Yes

*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 if the client type of the request is *CONTROL, 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.

HSM Server program code implementation: Yes
HSM Client program code implementation: No

*TIMEOUT

If an HSM request takes longer than the predetermined timeout period for Strategi in order to send a reply, Strategi will end the request and send as the reply Opcode *TIMEOUT. In essence, the server request has taken too long to perform, and therefore, has timed out. Since Strategi sends the reply of *TIMEOUT, an HSM Server should never attempt to send *TIMEOUT as a reply Opcode.

With advanced HSM Servers, the request timeout length can be set using the HSM Server Setting Option API (e.g., HSMSVROPT).

HSM Server program code implementation: No
HSM Client program code implementation: Yes

User-defined Opcodes

All non-reserved Opcodes are user-defined, predetermined at the time of Server design, and used by the client application for making requests. The only restrictions on naming are that the Opcode cannot be longer than ten characters and cannot start with an asterisk because Opcodes beginning with an asterisk are considered reserved for Strategi's specialized usage.

HSM Client Special Values

The following chart details the HSM Client Special Values used within an HSM client program:

Field	**Value**	**Meaning**
Keyword	CCSID	Specify CCSID for all HSM elements
	PRPCCSID	Specify CCSID for property elements (e.g., opcodes)
	RQSCCSID	Specify CCSID for request data element
	RPYCCSID	Specify CCSID for reply data elements
Rqs Server	*SYSTEM	Strategi System Services
Rqs Opcde	*PING	Request server identity
Rpy Opcode	*ERROR	Error occurred in HSM transport
	*NOSERVER	Server is not running
	*PING	Server response to *PING request
	*TIMEOUT	Reply not received within specified time

Client Option		Value
CCSID		Number from 0 - 65535, *HEX, *JOB
PRPCCSID		Number from 0 - 65535, *HEX, *JOB
RQSCCSID		Number from 0 - 65535, *HEX, *JOB
RPYCCSID		Number from 0 - 65535, *HEX, *JOB

HSM Client Special Value Notes

Any opcode reply made with an opcode beginning with '*' that is not an HSM Client Special Value will be rejected. Opcodes beginning with '*' are reserved for Strategi, and therefore, must be a Strategi-recognized opcode. It is permissible to use Strategi-defined opcodes in the context for which they are designed.

The required reply data format for *ERROR opcode is the following:

```
  HSM_BYTE  MSGIDN [  7]  /* First 7 positions are message ID number */
  HSM_BYTE  SPACE  [  1]  /* Blank space to separate ID and text     */
  HSM_BYTE  MSGTXT [200]  /* Message text                            */
```

CCSID Options

Automatic translation of request/reply data is done by HSM. The default CCSID is the CCSID of the job when the request is received or the reply is sent. Data is moved in its CCSID; all property elements are converted to the Strategi internal CCSID (CP1140, US with Euro).

Using server options xxxCCSID with *HEX (binary data) values are not guaranteed supportable between all WebCluster systems. CCSID option setting was first introduced with Strategi V1R8.

Strategi only supports pure Single-Byte Coded Charater Set (SBCS) and Double-Byte Coded Character Set (DBCS) code pages. Mixed character set code pages are not permitted. More information regarding Coded Character Set Identifiers (CCSIDs) can be found on IBM's websites (e.g., www.ibm.com). Specifically, the manual on "National Language Support" contains term definitions and tables of CCSIDs.

Chapter 8. Built-in Function Implementations

Custom Login

Strategi website zones can be configured to allow custom login user authentication pages rather than using the browser's default login dialogue box, as with *BASIC authentication.

With *CUSTOM authentication, one page within the *CUSTOM authenticated zone will be delivered without any authentication required (SSL requirements still apply where configured). The name of this page is specified by the Strategi Special Value LOGINURL, and the default value shipped with Strategi is "login". The value can be specified with or without an extension, depending upon whether different login page types are desired for different zones. One zone might deliver "login.htm" while another "login.shtm". The page identified as the login page will be used to log the user into the zone.

The benefit to having custom login pages is consistent site look-and-feel and ease of customer login.

Zone Configuration

The basic requirements for properly setting up a Strategi website zone for Custom Login involve creating a Strategi website zone and HTML page with FORM elements.

From the Strategi main menu (SGI),

1. Web Sites (option 6)
2. Work with Web Site Zones (option 12)
3. Create (F6)

- Enter "Zone Code"
- Enter "Subdirectory (within domain)"
- Enter "Text Description"
- Enter "Encoding For Text Resources"
- Enter *CUSTOM for "User Authentication"
- Enter "Default User Authority"
- Enter remaining parameters
- Press Enter

Once Strategi is restarted the new website zone authorities will take effect. The next step is to create the login HTML page with FORM elements capable of receiving user input.

HTML Page Requirements

The login page must contain an HTML form which makes a post. The action of the form must be a web page within the same zone, appending "?*LOGIN" to the target filename (e.g., action="main.htm?*LOGIN"). The form must also include two input-capable fields, *LOGINUSR and *LOGINPWD, representing Strategi Username and Passphrase, respectively.

These two form fields carry the required Strategi user information for authenticating to the Strategi webserver.

Example login web page:

login.htm

```
<HTML>
<BODY>
<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>
</HTML>
</BODY>
```

The targeted page, MAIN.HTM, will normally contain links that allow logging out.

main.htm

```
<HTML>
<BODY>
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>
</HTML>
</BODY>
```

These logout links can be included on any page within the zone. The logout link has a very specific structure. Notice that no actual html page is required in the href attribute to target, just a ".". Instead, two variable=value pairs are being sent on the http request.

The first variable=value pair, *SESSION=LOGOUT, tells the server to log the user out of the zone. The second one, *LOGOUTREDIRECT=/homepage.htm, sends the user to a different web page upon logout. ("/homepage.htm" is used for example purposes only.)

The redirect to '/homepage.htm' means that HOMEPAGE.HTM in the root of the website will be served. The redirection could target any valid page on the website. However, zone security requirements must still be met before Strategi will serve the web page.

A redirection or return to the login page will log the user out.

HSM Page Requirements

An associated HSM file is not required to support Custom Login. However, if HSM processing is desired, Strategi will perform HSM processing on the login page for a custom-defined zone. Initiate HSM processing by either specifying the login filename extension as SHTM or SHTML, or by creating an associated HSM file (e.g., login.hsm).

Strategi supports using HSM Special Values and HSM server requests via the login-associated HSM file.

Strategi will pass URL variables to your login page if HSM processing is required.

HSM File Upload

The Strategi web server has the ability to receive uploaded files. 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 iSeries Integrated File System (IFS). Once uploaded, Strategi's System Server Opcodes allow file information retrieval and manipulation.

Upload Implementation Example

For example, the HTML file "postimg.htm" is located in the IMAGE zone that is secured with *BASIC authentication. There are no restrictions on the type of PC file to upload, but we will use image files for this example.

postimg.htm

```
<HTML>
<BODY>
<FORM method="post" enctype="multipart/form-data"
  action="postimg.hsm">
New Image: <INPUT type="file" name="NewFile">
<INPUT type="submit" value="UploadNow"><BR>
<BR>
<INPUT type="hidden" name="NewFile.authorityzone" value="IMAGE"><BR>
<BR>
</FORM>
</BODY>
</HTML>
```

postimg.hsm

```
[DO]
```

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

The special value of *CURRENT is allowed for the **authorityzone** property. This special value, which is case sensitive, allows for securing with a zone located in a non-active (*SUSPENDED) or active website via aliasing. Its use is not limited to aliased zones, but provides added flexibility if using a Strategi zone on a non-active website for file upload purposes only. Prior to this, all zones used for securing the uploaded file required the zone be on an active website. The input tag will look like the following:

```
<INPUT type="hidden" name="NewFile.authorityzone" value="*CURRENT">
```

Upon signing into the IMAGE zone, loading the page, selecting the file and clicking "UploadNow", the selected file will be sent to the iSeries. 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 iSeries, it does not end up in the zone containing the upload web page. 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 iSeries 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 web page by referring to "*PUBLISH/0000000001". More likely, you would have a variable FileHandle and refer to it like this:

```
<IMG src="*PUBLISH/<HSM name="FileHandle">">
```

How does one access the file handle in the first place? Well, in the example HTML, the line <INPUT type= "file" name="NewFile"> means that when the form is submitted, the variable "NewFile.handle" will be available within HSM. This NewFile.handle could be then used on a following web page, 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.

postimg.htm

```
<HTML>
<BODY>
<FORM method="post" enctype="multipart/form-data"
  action="postimg.hsm">
New Image: <INPUT type="file" name="NewFile">
<INPUT type="submit" value="UploadNow"><BR>
<INPUT type="hidden" name="NewFile.authorityzone" value="IMAGE"><BR>
</FORM>
Last Image: <IMG src="*PUBLISH/<HSM name="FileHandle" escape="URL">">
</BODY>
</HTML>
```

postimg.hsm

```
[DO]
condition_variable= NewFile.handle
condition_criteria= DEFINED
assign_field_value= FileHandle, NewFile.handle
```

Upload Error Handling

Beginning with Strategi V1R4, most upload errors no longer fail the HTTP request with HTTP Codes 524 or 403, but continue with HSM processing with error details set. The error details consist of an error code and associated error text.

In order to make error handling simplified, a numeric error code which is easily tested in the HSM Resource file is delivered separately from error text describing the error. This allows conditional processing based upon error code while still delivering user-friendly textual information.

If the **name** attribute value of the of the INPUT tag is "NewFile", then on an error HSM variables will be:

NewFile.errorcode	with the numeric code
NewFile.errortext	with the descriptive text
NewFile.handle	will not be defined

Possible errors that can be returned, with replaceable text enclosed in '<>':

01 – File empty or not found - <filvar>
02 – Unknown error
03 – No authority zone variable - <filvar.authorityzone>
04 – Authority zone not valid - <uplzoncod>
05 – Access to upload area denied
06 – Upload file open failed
07 – Authority zone variable blank - <filvar.authorityzone>

where

<filvar> is replaced by the **name** attribute value of the file INPUT tag
<uplzoncod> is replaced by the upload authority zone code

If no file name is entered in the HTML, neither NewFile.handle nor NewFile.errorcode will be set. The is no distinguishing between an empty file and an invalid file name. The HSM file could have the following:

postimg.hsm

```
[DO]
condition_variable= NewFile.errorcode
condition_criteria= EQUAL
condition_compare_value= 01
substitute_url= uploadFileInvalid.htm

[DO]
condition_variable= NewFile.errorcode
condition_criteria= DEFINED
substitute_url= uploadFileError.htm

[DO]
condition_variable= NewFile.handle
condition_criteria= NOTDEFINED
substitute_url= uploadFileNotEntered.htm
```

uploadFileError.htm

```
<HTML>
<BODY>
<H1>Upload File Error</H1>
Error Text: <HSM name="NewFile.errortext">
</BODY>
</HTML>
```

File Upload Operation and Property Opcodes

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.

CHGPROH

Changes the value of properties for a given file when passed its handle. If the property does not exist, it is created. The return Opcode and values are the same as those sent, or ERROR if handle does not exist.

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= CHGPROH
assign_field_value=PropertyName1,"Object Color"
assign_field_value=PropertyValue1,"green"
start_length_field=   1,  10,    Handle
start_length_field=   *,  20,    PropertyName1
start_length_field=   *, 128,    PropertyValue1
```

Error Codes:

```
2603 - Property is not defined
```

CPYHFIL

Copies a published file or files or a website file or files to a specified path in a zone on a Strategi website.

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= CPYHFIL
start_length_field=   1,  10,    "*HANDLE"
start_length_field=   *,  10,    HandleSrcFile
start_length_field=   *,  10,    HandleTgtFile

or

[SERVER REQUEST]
Server= *SYSTEM
opcode= CPYHFIL
start_length_field=   1,  10,    "*WEBZONPTH"
start_length_field=   *,  10,    HandleSrcFile
start_length_field=   *,  15,    WebsiteCode
start_length_field=   *,  15,    ZoneCode
start_length_field=   *,1024,    ToRelativePathWithinZone
start_length_field=   *, 256,    AlternateFileName

[REPLY]
opcode= CPYHFIL
start_length_field=   1,  10,    Size
start_length_field=   *,  14,    ModifiedDateTime
start_length_field=   *,  14,    StatusDateTime
start_length_field=   *,  14,    AccessDateTime
start_length_field=   *,  11,    ObjectType
start_length_field=   *,   5,    CodePage

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
20H1 - Relative path not permitted in sub path field
20H2 - Internal Strategi error.
20H3 - Specified website not defined.
20H4 - Specified zone not defined within website.
20H5 - Access to zone denied (user requires *WRITE access)
20H6 - Invalid path type specified.
20H7 - Source file not found on disk (handle points to non-existent
       file).
20H8 - Failed to copy file to alternate name.
20H9 - Target file not present after copy (should never happen).
```

DLTHFIL

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

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= DLTHFIL
start_length_field=   1,  10,    "*HANDLE"
start_length_field=   *,  10,    HandleSrcFile
start_length_field=   *,  10,    HandleTgtFile

or

[SERVER REQUEST]
Server= *SYSTEM
opcode= DLTHFIL
start_length_field=   1,  10,    "*WEBZONPTH"
start_length_field=   *,  10,    HandleSrcFile
start_length_field=   *,  15,    WebsiteCode
start_length_field=   *,  15,    ZoneCode
start_length_field=   *,1024,    RelativePathWithinZone

[REPLY]
opcode= DLTHFIL
start_length_field=   1,  10,    FileSize
start_length_field=   *,  14,    ModifiedDateTime
start_length_field=   *,  14,    StatusDateTime
start_length_field=   *,  14,    AccessDateTime
start_length_field=   *,  11,    ObjectType
start_length_field=   *,   5,    CodePage

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
20H1 - Relative path not permitted in sub path field
20H2 - Internal Strategi error.
20H3 - Specified website not defined.
20H4 - Specified zone not defined within website.
```

```
20H5 - Access to zone denied (user requires *WRITE access)
20H6 - Invalid path type specified.
20H7 - Source file not found on disk (handle points to non-existent
       file).
20H8 - Failed to delete file.
```

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=   *,1024,    FullPath
   * FullPath = complete IFS path, including leading "/"

or

[SERVER REQUEST]
Server= *SYSTEM
opcode= FILEINFO
start_length_field=   1,  10,    "*WEBZONPTH"
start_length_field=   *,  15,    WebsiteCode
start_length_field=   *,  15,    ZoneCode
start_length_field=   *,1024,    RelativePath
* RelativePath = relative path, no leading "/"

or

[SERVER REQUEST]
Server= *SYSTEM
opcode= FILEINFO
start_length_field=   1,  10,    "*HANDLE"
start_length_field=   *,  10,    Handle

[REPLY]
opcode= FILEINFO
start_length_field=   1,  10,    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=   *,  14,    ModifiedDateTime
   * CCYYMMDDHHMMSS datetime (contents) last modified (st_mtime)
start_length_field=   *,  14,    StatusDateTime
   * CCYYMMDDHHMMSS datetime status changed (st_ctime)
start_length_field=   *,  14,    AccessDateTime
   * CCYYMMDDHHMMSS datetime last accessed (st_atime)
start_length_field=   *,  11,    ObjectType
   * Type of object, always "*STMF" for *HANDLE requests.
start_length_field=   *,   5,    Codepage
start_length_field=   *,   5,    ImageWidth
   * Image width (if known for .gif .jpg .jpeg files, else zero)
start_length_field=   *,   5,    ImageHeight
   * Image height (if known for .gif .jpg .jpeg files, else zero)
```

```
[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
00G1 - Invalid path type requested
00G2 - File not found
00G3 - Website list not generated (should not actually happen)
00G4 - Website not defined in Strategi
00G5 - Zone not defined in Strategi
00G6 - Transfer tracking reference number not found
```

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,     PropertyName1
start_length_field=   31,20,     PropertyName2
start_length_field=   51,20,     PropertyName3

[REPLY]
opcode= GETPROH
start_length_field=   1,  10,    Handle
start_length_field=   *, 128,    PropertyValue1
start_length_field=   *, 128,    PropertyValue2
start_length_field=   *, 128,    PropertyValue3

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

MOVHFIL

Moves a published file or files or a website file or files to a specified path in a zone on a Strategi website.

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= MOVHFIL
start_length_field=   1,  10,    "*HANDLE"
start_length_field=   *,  10,    HandleSrcFile
start_length_field=   *,  10,    HandleTgtFile

or
```

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= MOVHFIL
start_length_field=   1,  10,    "*WEBZONPTH"
start_length_field=   *,  10,    HandleSrcFile
start_length_field=   *,  15,    WebsiteCode
start_length_field=   *,  15,    ZoneCode
start_length_field=   *,1024,    ToRelativePathWithinZone
   * or you can use the following ...
start_length_field=   *, 256,    AlternateFileName

[REPLY]
opcode= MOVHFIL
start_length_field=   1,  10,    FileSize
start_length_field=   *,  14,    ModifiedDateTime
start_length_field=   *,  14,    CreationDateTime
start_length_field=   *,  14,    AccessDateTime
start_length_field=   *,  11,    ObjectType
start_length_field=   *,   5,    CodePage

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
20H1 - Relative path not permitted in sub path field
20H2 - Relative path not permitted in alternative name field
20H3 - Internal Strategi error.
20H4 - Specified website not defined.
20H5 - Specified zone not defined within website.
20H6 - Access to zone denied (user requires *WRITE access)
20H7 - Invalid path type specified.
20H8 - Source file not found on disk (handle points to non-existent
       file).
20H9 - Failed to rename file to alternate name.
20HA - Target file not present after rename (should never happen).
```

SETIFSAUT

Sets standard authority and ownership for Strategi IFS objects. When publishing files to Strategi via HSM, and subsequently moving them to another location in a website or IFS location, the automated process that performs the move will become the object owner. SETIFSAUT allows a change in object ownership such that Strategi (e.g., SGIOBJOWN) becomes the owner and is able to perform normal Strategi-related operations. SETIFSAUT should be used on any object deposited via file upload into a Strategi website to ensure Strategi jobs have the necessary authority to perform required operations.

Use restrictions:

- The current object owner must be a profile that SGIOBJOWN has authority to
- Relative path elements (e.g., '..') are not allowed within the path name

- Paths beginning with '/Q' (e.g., '/QSYS.LIB') or '/' are not allowed

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= SETIFSAUT
start_length_field=   1,1024,    PathOfIFSObject

[REPLY]
opcode= DONE

[REPLY]
opcode= ERROR
start_length_field=   1,9999,    ErrorMsg
```

SETPROH

Creates a property for a given file when passed its handle. 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=PropertyName1,"Object Color"
assign_field_value=PropertyValue1,"red"
start_length_field=   1,  10,    Handle
start_length_field=   *,  20,    PropertyName1
start_length_field=   *, 128,    PropertyValue1

[REPLY]
opcode= SETPROH
start_length_field=   *,  10,    Handle
start_length_field=   *,  20,    PropertyName1
start_length_field=   *, 128,    PropertyValue1

[ERROR]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
2606 - Handle is not defined
```

Password Maintenance

Strategi's password maintenance System server Opcode allows a user to conveniently change their Strategi passphrase without requiring 5250 emulation access. By placing an HTML form and supporting HSM file into an authenticated zone, the System server will change the Strategi user's password by requiring the user to supply their existing password and a new one. The new one is entered twice, once each into two separate input fields, on the page for confirmation purposes.

It is not required USRPWD be invoked from an authenticated zone, but for security and ease of user interface purposes it is recommended. The user number is available in an authenticated zone via the HSM special value of *USER.NUMBER.

Password Maintenance Opcodes

USRPWD

Sets a Strategi user's password provided the Strategi user number and existing password are submitted on request. If the OptionalUserNumber is blank or zero, the logged on user number from the authenticated zone is used.

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= USRPWD
start_length_field=   1, 40,     CurrentPassword
start_length_field=   *, 40,     NewPassword
start_length_field=   *, 40,     NewPasswordConfirm
start_length_field=   *,  9,     OptionalUserNumber
start_length_field-   *, 40,     OptionalUserName

[REPLY]
opcode= DONE
start_length_field=   1,   4,    ErrorCode
start_length_field=   *,  96,    ConfirmationText
start_length_field=   *,   6,    DaysPasswordValid -or- *NOMAX

[REPLY]
opcode= FAIL
start_length_field=   1,   3,    ErrorCode
start_length_field=   *,   1,    Blank
start_length_field=   *,  100,   ErrorText
- or -
start_length_field=   1, 100,    CodedReasonForFailure

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes:

```
0003 - User number is not on file
9997 - Request length is invalid (too short)
```

Example usage of Opcode USRPWD:

usrpwd.htm

```
<HTML>
<BODY>
<FORM method="post" action="usrpwd.hsm">
Current Password
  <INPUT type="password" name="CurrentPassword" size="40"
```

```
    maxlength="40"><BR>
New Password
  <INPUT type="password" name="NewPassword" size="40"
    maxlength="40"><BR>
New Password (Confirm)
  <INPUT type="password" name="NewPasswordConfirm" size="40"
    maxlength="40"><BR>
<INPUT type="submit" name="PasswordSubmit" value="Change
    Password"><BR>
</FORM>
</BODY>
</HTML>
```

usrpwd.hsm

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= USRPWD
condition_variable=               PasswordSubmit
condition_criteria=               DEFINED
start_length_field=   1, 40,      CurrentPassword
start_length_field=   *, 40,      NewPassword
start_length_field=   *, 40,      NewPasswordConfirm

[REPLY]
opcode= DONE
start_length_field=   1, 100,     ConfirmationText
substitute_url=                   usrpwddone.htm

[REPLY]
opcode= FAIL
start_length_field=   1, 100,     ConfirmationText
substitute_url=                   usrpwdfail.htm
```

Pushfeed

Strategi's Pushfeed allows a user to download files and printouts that have been sent to them directly from the Strategi web server, 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.iseries.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.iseries.address)/(WebsiteZone)/*PUSHFEED/
       (StrategiUserNumber)/(FileNumber)/(FileName)
```

The "WebsiteZone" 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. “StrategiUserNumber” is the number of the user logged in. “File Number” is the reference number of the pushed file, and “FileName” 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.

Pushfeed Opcodes

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=   *,   3,    BeginningStatus
   * Beginning Status to include: "AVL"=Available, "XDL"=Deleted,
   * "RTV"=Retrieved "APN"=Pending, "ALL"=All Files
start_length_field=   *,   3,    EndingStatus
   * Ending Status to include.
start_length_field=   *,   10,   ReferencePosition
   * Where in the file list to begin
start_length_field=   *,   1,    ScrollDirection
   * Scrolling Direction "F" for forward, "B" for back
start_length_field=   *,   4.0,  MaxNbrOfEntries
   * The maximum number of entries to be returned

[REPLY]
opcode= LSTPUSH
start_length_field=   1,   4,    NbrOfEntriesReturned
start_length_field=   *,   1,    MoreAfter
   * Y if more entries exist for scroll forward, else N
start_length_field=   *,   1,    MoreBefore
   * Y if more entries exist for scroll back, else N
start_length_field=   *,  20x10, FileReference
start_length_field=   *,  20x9,  User
   * User (currently all the same)
start_length_field=   *,  20x3,  Status
start_length_field=   *,  20x1,  Priority
start_length_field=   *,  20x4,  NbrOfTimesRetrieved
start_length_field=   *,  20x40, Description
start_length_field=   *,  20x40, Origin
start_length_field=   *,  20x64, FileName
start_length_field=   *,  20x1,  TransferType
```

```
start_length_field=   *,  20x64, OverridingMimeType
start_length_field=   *,  20x10, Size
start_length_field=   *,  20x14, FileDateTime
start_length_field=   *,  20x14, CreateDateTime
start_length_field=   *,  20x14, QueueDateTime
start_length_field=   *,  20x14, SentDateTime
start_length_field=   *,  20x14, LastActionDateTime
start_length_field=   *,  20x14, DeleteDateTime

[REPLY]
opcode= *ERROR
start_length_field=   1,   7,    SysMsgID
start_length_field=   *,   1,    Blank
start_length_field=   *, 200,    ErrorText
```

Error Codes:

```
SVR0002 - User is not logged in, or different user specified than
          that logged in.
```

DLTPUSH

Deletes the requested file. Because of the array size, the maximum number of references to delete is 20.

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= DLTPUSH
start_length_field=   1,      9, *USER.NUMBER
start_length_field=   *,      4, NbrOfReferencesToDelete
start_length_field=   *,  20x10, FileReferencesToDelete

[REPLY]
opcode= *ERROR
start_length_field=   1,   7,    SysMsgID
start_length_field=   *,   1,    Blank
start_length_field=   *, 200,    ErrorText
```

Error Codes:

```
SVR0002 - User is not logged in, or different user specified than
          that logged in.
SVR0003 - Too many references specified (greater than 20).
SVR0004 - Reference count was non-numeric.
```

Note that the iSeries 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.

Website Information

Website Information Opcodes

Strategi's website information opcodes allow a user to retrieve various pieces of information pertaining to their website and website IFS directories.

GETZONSTA

Gets the website zone state and Strategi user access name of a Strategi website zone for a given website domain and zone code.

Possible reply ZoneState values:
0 – Never checked out
1 – Checked in
2 – Checked out
3 – Checking in process
4 – Checking out process

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= GETZONSTA
start_length_field=   1, 15,     WebsiteCode
start_length_field=   *, 15,     ZoneCode

[REPLY]
opcode= DONE
start_length_field=   1,  1,     ZoneState
start_length_field=   *, 40,     AccessName

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
0019 - Could not retrieve zone state (zone not defined).
9997 - Request length too short.
```

LSTWEBSIT

Returns a list of all defined websites. It can be used in conjunction with a web form to provide a means for the user to select a website for an operation.

```
[SERVER REQUEST]
Server= *SYSTEM
opcode= LSTWEBSIT
start_length_field=  1,   1,     IncludeSuspendedWebsite
   * Possible returned values are 0=No and 1=Yes
```

```
[REPLY]
opcode= DONE
start_length_field=  1,      4,  WebsiteCount
start_length_field=  *, 500x16,  WebsiteCodes

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

ZONPTH
Returns the IFS path of a nominated Strategi zone.

```
[SERVER REQUEST]
Server= *SYSTEM
Opcode= ZONPTH
start_length_field=   1,   10,   ZoneType
   * WEBSITE - Asking for a website zone path
start_length_field=   *,   50,   RelZoneDomain
   * WEBSITE - The website code
start_length_field=   *,   50,   ZoneCode
start_length_field=   *, 1000,   SubDir

[REPLY]
opcode= DONE
start_length_field=   1, 9999,   IFSPath

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
0009 - Zone type not recognized
0010 - Website code not found on file
0011 - Zone code not found on file
```

SYSPTH
Returns a complete IFS path given specific Strategi data.

```
[SERVER REQUEST]
Server= *SYSTEM
Opcode= SYSPTH
start_length_field=   1,   10,   PathType
   * WEBSITE – A website root path
   * WEBSITENA – A website root path without resolving the alias
start_length_field=   *,   50,   RelPathCode
   * WEBSITE – The website code
   * WEBSITENA – The website code
start_length_field=   *, 1000,   SubDir
   * To be appended to the final path
```

```
[REPLY]
opcode= DONE
start_length_field=   1, 9999,   IFSPath

[REPLY]
opcode= ERROR
start_length_field=   1,  30,    RequestOpcode
start_length_field=   *,   4,    ErrorCode
start_length_field=   *,  80,    ErrorText
```

Error Codes

```
0009 - Path type code not recognized
0010 - Invalid website code
```

Chapter 9. WebCluster

Distributed HSM Servers (DHSM)

Distributed HSM (DHSM), allows HSM 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 that 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:

```
                  LOCAL
========================================
=                                      =
=  ----------     *STRATEGI <SGI USER> =
= |  Client  |    |                    =
= |----------|    |      ------------  =
= |HSM SRVPGM| <-USRQ-> | HSM SRVPGM | =
=  ----------           |------------| =
=                       | Relay PGM  | <------------
=                        ------------  =            |
=                                      =            |
========================================            |
                                                    |
                                                  TCP/IP
                  HOST                              |
========================================            |
=                                      =            |
=                        ------------  =            |
=                       | Agent PGM  | <------------
=  ----------           |------------| =
= |HSM SRVPGM| <-USRQ-> | HSM SRVPGM | =
= |----------|    |      ------------  =
= |  Server  |    |                    =
=  ----------     *PEER <SYS CODE>     =
=                                      =
========================================
```

DHSM Setup Steps

There are four basic steps required to successfully setup DHSM:

- 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)

Authentication & Security

There are three key fields used to identify a peer system: serial number, system name and Strategi library. All three components are used for authentication purposes and allows Strategi's WebCluster to run in an LPAR environment.

The serial number of the peer system must match RTVSYSVAL QSRLNBR; the system name must match RTVNETA SYSNAME; the Strategi library is the library in which Strategi is currently installed. The Strategi software will retrieve these values and pass it to the other end to identify itself. This combination of serial number, system name and Strategi library is analogous to a User's access name. Once the values are 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 local-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 three 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.

Chapter 10. Additional Topics

Resources

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

SGIEXAMPLE

Contained within the Strategi-installed library is the source physical file SGIEXAMPLE, which contains example, skeleton source members to use for ease in creating HSM server programs. Examples exist in different programming languages to suit a variety of programming skill sets. Choose the one that best meets the business requirements for your project.

SGITOOL

Contained within the Strategi-installed library is the source physical file SGITOOL, which contains public tools to use for ease in creating Java HSM Servers. Command help text panels and program source are included for customization and modification.

Support Website

ADVANCED BusinessLink's Support website [http://support.businesslink.com/] contains Technical Support Bulletins, product upgrades, downloads (including product manuals), and contact information. The download area of the BusinessLink support website will contain the source for various demonstration / example applications.

HSM Server Packaging & Installation Commands

The PKGHSMSVR and INSHSMSVR are Strategi commands that allow HSM developers to distribute HSM applications to other iSeries running Strategi. PKGHSMSVR is used to prepare a HSM application for distribution on the source iSeries while INSHSMSVR is used to install the HSM application on the destination iSeries. 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

PKGHSMSVR command parameters:

```
  SVRNAM  name of server to package
  LIB     library where server program resides
  DIR     website code and subdirectory of IFS objects
  DEV     iSeries 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 (e.g., V4R2M0).

- All objects must be owned by a profile that the user who installs the application on the destination iSeries 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 iSeries.

- 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 iSeries 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

INSHSMSVR command parameters:

```
  SVRNAM  name of server to install
  LIB     library for server objects
  DIR     website code and subdirectory of IFS objects
  DEV     iSeries 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; however, it might require a repackage on the source system depending on the severity.

Additional Questions

Listed below are a few questions with associated answers that might help in solving a particular HSM programming need:

Q: On a reply, can data sent back to a page to a substitute URL, be made to check a checkbox in the HTML?
A: To check a checkbox, output the string "checked" as the value of an HSM reply variable (e.g., SomeVar).

For example:

```
  start_length_field=11,7, SomeVar
```

and then the HTML can be:

```
  <INPUT type="checkbox" name="SomeName" <HSM name="SomeVar">>
```

which will resolve to:

```
  <INPUT type="checkbox" name="SomeName" checked>
```

Please note that this is “technically” invalid HTML, because the <HSM> tag has been nested inside the <INPUT> tag. However, for Strategi’s HSM this is acceptable because the web 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 to differentiate the variable name from its value, but in an application it is natural for them to be the same.

Q: 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?
A: 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 to differentiate the variable name from its value, but in an application it is natural for them to be the same.

Appendix A. Converting Old Servers

Overview

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 (iSeries) 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 previously outlined.

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 more "maintainable" program
- Client type provided 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       4P 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).`

Index