QACOM Reference Guide for the .NET Environment

Table of Contents

Using Verastream Bridge Integrator in the Microsoft .NET Environment

When using Verastream Bridge Integrator, you must first import a reference to the VBI DLL. If you are securing your connection to the host, your client application will load faster if you also import a reference to the VBI XmlSerializers DLL.

To declare the Bridge API in a VB application:

```
' Don't forget to import the reference to VBI.dll and
VBI.XmlSerializers.dll

Dim bridgeAPI As VBI.BridgeAPI

bridgeAPI.ConnectBridge(…)

…
```

You can simplify the declaration of the variable `bridgeAPI` by adding VBI to your list of imported namespaces:

```
Imports VBI

Dim bridgeAPI As BridgeAPI

bridgeAPI.ConnectBridge(…)

…
```

Securing Host Communications

To enable a secure connection (encrypted communication to a Web service on the host using SSL over HTTPS), follow these simple steps:

1. Import required certificated into the Windows Certificate Manager.
2. Import a reference to the .NET assembly `System.Web.Services`.
3. Use one of three possible methods (listed below) to configure a secure implementation used by the Bridge API .
4. Add your certificates to certificate collection on the Web Service.

Methods to configure the implementation used by the Bridge API

- Recommended. Use a version of the `BridgeAPI` constructors that has a `secureConnection` argument. Pass in a value of `True`.
- Use the new static `BridgeAPIFactory` methods to configure the BridgeAPI for a SOAP implementation that uses a secure connection.
- Set the BRIDGE_DEFAULT_SECURE_CONNECTION system environment variable to `True`.
  There are two system environment variable you can use to configure the implementation:
  - BRIDGEAPI_DEFAULT_IMPL – Set to either "binary" or "SOAP". By default, the internal Bridge API implementation in binary (non-secure). However, if you set this environment variable to "SOAP", it will use the SOAP implementation and connect to the Web service on the host.
  - BRIDGE_DEFAULT_SECURE_CONNECTION – Set to "true" to secure the connection with SSL. Currently, only the SOAP implementation can be

secured, which forces the SOAP implementation. This is the only system environment variable you need to set. Using a non-secured SOAP connection is not recommended, but can be used temporarily if security has not yet been configured on the host.

Of these three methods of configuration, the first approach is the simplest. The second provides a fine-grained control over the implementation configuration, which can be useful during the development phase when the configuration with the host is not stable. Finally, system environment variables can be useful if you need to control the implementation used from outside of your application.

Sample Code for Establishing a Working Connection

The following code gives you a working connection if you have the appropriate certificate imported into the Windows Certificate Manager:

```
Imports System.Security.Cryptography.X509Certificates
Imports VBI

Dim store As X509Store = New X509Store(StoreName.Root,
StoreLocation.CurrentUser)
Dim collection As X509Certificate2Collection
Dim bridgeAPI As BridgeAPI

store.Open(OpenFlags.ReadOnly)
collection = New X509Certificate2Collection()
collection.Find(X509FindType.FindByIssuerName,
"MyCertificateIssuerName", True)
store.Close()

bridgeAPI = New BridgeAPI(True)

bridgeAPI.GetWebService().ClientCertificates.AddRange(collection)
bridgeAPI.ConnectBridge(…)
…
```

You can also obtain a particular implementation of the Bridge API using `BridgeAPIFactory`:

```
bridgeAPI = BridgeAPIFactory.createBridgeAPI
(BridgeAPIFactory.ImplTypeEnum.SOAP, True)
```

Or, you can rely on a system environment variable to configure the connection:

```
Environment.SetEnvironmentVariable
("BRIDGEAPI_DEFAULT_SECURE_CONNECTION", "true") 'Or set this
variable in the system environment itself
bridgeAPI = New BridgeAPI()
```

Configuring Bridge API Code Pages

The `BridgeAPI` class has six overloaded constructors. One of the common arguments, `secureConnection`, is described in the previous section. The other arguments have to do with code page configuration. The argument, `hostCodePage` (probably the most important), is needed when connecting to a host that is configured to use a code page other than 037 (the default).

Supported host code pages:

037	US English, Portuguese
1140	US English, Portuguese (Euro)
1141	German, Austrian (Euro)
1142	Danish, Norwegian (Euro)
1143	Finnish, Swedish (Euro)
1144	Italian (Euro)
1145	Spanish (Euro)
1146	UK English (Euro)
1147	French (Euro)
1149	Iceland (Euro)
273	German, Austrian
277	Danish, Norwegian
278	Finnish, Swedish
280	Italian
284	Spanish
285	UK English
297	French
871	Iceland

The `clientCodepage` argument defaults to the encoding used for the system's current ANSI code page – usually 1252 on a US Windows machine.

The encoding is determined by passing the value of `clientCodepage` into the `System.Text.Encoding.GetEncoding` static method. See .NET documentation on the `System.Text.Encoding.GetEncoding` class for information on allowable values.

When using a secure connection (or a non-secure SOAP connection), the `hostCodePage` and `clientCodepage` arguments are ignored. The host code page value is determined from the terminal facility. The client code page is not used since characters are sent to the host in Unicode.

Optimizing High Performance 3270 Solutions

When building an application that makes use of the Web service, use care to minimize the number of APIs used to build a given solution. Each API formats a TCP/IP request packet and makes a round trip to the CICS region running on the target host. You can accelerate your applications if some of the TCP/IP packet trips can be eliminated. The greatest level of performance can be achieved by using the following optimization tips.

SendKeys Optimization

Your application will undoubtedly make use of several SendKeys() API to drive the host CICS application to the desired screens. Although the programming effort of coding these keystrokes as a series of distinct API seems easier, by "stacking" multiple API into a single invocation, many 3270 exchanges can be accomplished with just one trip to the host.

For example, consider an application that connects to a given CICS region, enters a transaction 'APPL' to determine how many crates of apples are in storage, makes a menu selection "1" to designate Golden Delicious apples, presses <ENTER>, enters the crate size desired (12 pound) into the second field of the 3270 screen, and presses <ENTER> to go to the confirmation screen.

By coding with distinct API, the solution might look something like this:

```
ConnectBridge("nnn.nnn.nnn.nnn", 2317, 2, "MYUSERID", "PASSWD") 'Connect to CICS region
SendKeys("APPL@E")               'Start the 'Apples' transaction
SendKeys("1@E")                  'Select 'Golden Delicious'
SendKeys("@T")                   'Tab to the second field on the screen'
SendKeys("12@E")                 'Enter '12 pound crate' in 2nd field
```

Note that five API are used in the application so far, and consequently five TCP/IP request packets will be exchanged with the host to invoke the various API services illustrated above.

Now let's look at the same transaction invoked by "stacking" the SendKeys.

To implement this interaction, all of the keystrokes from the three distinct SendKeys calls will be stacked up in a single SendKeys call as follows:

```
SendKeys("APPL@E1@E@T12@E")
```

This will sign on, start the Apples transaction, Select Golden Delicious, specify 12 pound crate, and return with the 3270 screen parked on the confirmation screen. The application will then be free to harvest data from the confirmation screen (for instance, returning a confirmation number to the caller). The application could then issue a SendKeys to exit the application, and issue a Disconnect, freeing the session for another client.

The savings in terms of network time is substantial, and worth the analysis effort of the developers.

GetString Optimization

When retrieving data from many 3270 fields on a single screen, it may be worth issuing one GetString(sess, 0, 0, 1920, scrbuff) to retrieve the entire 3270 screen into your local buffer. Once all of the data for the screen is contained in the application's local buffer, carving multiple fields out of that 3270 screen buffer becomes a fairly simple programming task. The code might look like the following to return four 20 byte fields from column 15 of rows 10, 11, 12, 13 and 14 of a given screen.

```
GetString(10, 15 400, scrbuff)     'Get 5 lines worth of 3270 data

Field1 = Mid(scrbuff, 1, 20)       'Get the data from field at (10,15)
Field1 = Mid(scrbuff, 81, 20)      'Get the data from field at (11,15)
Field1 = Mid(scrbuff, 161, 20)     'Get the data from field at (12,15)
Field1 = Mid(scrbuff, 241, 20)     'Get the data from field at (13,15)
Field1 = Mid(scrbuff, 321, 20)     'Get the data from field at (14,15)
```

Again, with a bit more work, the application is saved four additional trips across the network. During the design of an application, you can identify many ways to reduce the number of API trips to the host, and with a reasonable amount of analysis; you can create the fastest application possible for your CICS applications.

Make sure that you need to verify where you are before asking for a GetString. Often, given the knowledge of your application, it is unnecessary to confirm several elements on each and every intermediate screen. You can confirm once, at the end of a chain of host screens,

that the proper destination has been reached without the involvement multiple GetString API along the way.

Combined Optimizations

By combining lessons learned from SendKeys optimization and GetString optimization, further improvement is attainable. Where appropriate, you can further reduce the total network latency by combining a concatenated SendKeys with an optimized GetString into a single SendKeysGetString call. With this approach, the concatenated SendKeys and the subsequent GetString are executed in a single trip to the host. Using the examples from the previous optimization sections, we would issue the following high performance call:

```
SendKeysGetString("APPL@E1@E@T12@E", 10, 15, 400, scrbuff)
```

Other calls allowing multiple input/outputs are SendKeysGetAllFieldInformation and X3270Screen. See reference information for those APIs in this guide.

SetFieldValue and GetFieldValue Optimization

When using a CICS application that makes use of BMS mapsets, the SetFieldValue and GetFieldValue APIs allow updating and retrieving fields in the 3270 session without resorting to sending "Row/Column/Length" information in a PutString or GetString API.

When you use SetFieldValue, the symbolic name of the field and the field's value are sent in the SetFieldValue API. There is an overloaded SetValue function that you can use to pass in an array of symbolic fieldnames along with a corresponding array of field values again all in one trip over the network (see SetFieldValue). Using this overloaded function will greatly improve the performance of your application.

When using GetFieldValue, the symbolic name of the field and the field's value are sent in the GetFieldValue API. Note that if many fields are to be retrieved from the host session, you may want to consider using the GetAllFieldInformation or GetScreenInformation API to retrieve them all from the host over one network trip, and then pick the desired target field name, value, row, column, attribute, length, color or highlighting information out of the arrays of information returned by the GetAllFieldInfomation or GetScreenInformation API. This approach reduces network traffic. See reference information for those APIs in this guide.

QACOM Methods

All supported QACOM methods return a zero on success, or an error code used to identify the specific problem.

ChangePassword

This method changes the password for a specified existing userid and password combination.

Syntax

```
ChangePassword hostIP, hostPort, UserId, Password, newPassword
```

Prerequisites

None

Parameters

Name	Description	Type
hostIP	Text format IP address of the target host CICS region.	String
hostPort	Port corresponding to the target CICS region.	Integer
UserId	The user ID for the host data engine to use with the External Security Manager.	String
Password	The password for the host data engine to use with the External Security Manager.	String
newPassword	The new password for the host data engine to use with the External Security Manager.	String

ConnectBridge

This method obtains a session state context from the host data engine.

Syntax

```
ConnectBridge hostIP, hostPort, terminalModel, UserId, Password
```

Prerequisites

None

Parameters

Name	Description	Type
hostIP	Text format IP address of the target host CICS® region.	String
hostPort	Port corresponding to the target CICS® region	Integer
terminalModel	3270 terminal model type 2 Model 2 3 Model 3 4 Model 4 5 Model 5	Integer
UserId	The user ID for the host data engine to use with the External Security Manager on behalf of this session. If no user ID is desired, use a null string or spaces.	String
Password	The password for the host data engine to use with the External Security Manager on behalf of this session. If no user ID is desired, use a null string or spaces.	String

ConnectBridge(Overloaded Methods)

This overloaded method is used to request additional and specific characteristics to be associated with this request for a session state context from the host data engine.

Syntax

```
ConnectBridge hostIP, hostPort, terminalModel, UserId, Password,
terminalPool

ConnectBridge hostIP, hostPort, terminalModel, UserId, Password,
terminalPool, networkName

ConnectBridge hostIP, hostPort, terminalModel, UserId, Password,
terminalPool, networkName, terminalFacilityLike
```

Prerequisites

None

Parameters

Name	Description	Type
hostIP	Text format IP address of the target host CICS® region.	String
hostPort	Port corresponding to the target CICS® region	Integer
terminalModel	3270 terminal model type 2 Model 2 3 Model 3 4 Model 4 5 Model 5	Integer
UserId	The user ID for the host data engine to use with the External Security Manager on behalf of this session. If no user ID is desired, use a null string or spaces.	String
Password	The password for the host data engine to use with the External Security Manager on behalf of this session. If no user ID is desired, use a null string or spaces.	String
terminalPool	Identifies the pool of Terminal IDs to search in order to specify a Terminal ID that is not currently in use.	String
networkName	Used to identify the specific network name to use with this session. The network name specified must conform to the VTAM rules for network name.	String
terminalFacility Like	Used to override the "virtual facility like" configured on the host for the terminalModel.	String

Disconnect

This method disconnects the session, releasing the session state context from the host data engine.

Syntax

```
Disconnect
```

Prerequisites

None

FindString

This method attempts to locate a given string on the current screen, beginning at a specified row and column position.

Syntax

```
FindString inScreen, inString, inStartRow, inStartColumn, inFlags,
outRow, outColumn
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
inString	The string to search for on the current screen.	String
inStartRow	The row position at which to start the search. This parameter must be greater than or equal to 1.	Integer
inStartColumn	The column position at which to start the search. This parameter must be greater than or equal to 1.	Integer
inFlags	Flags that control the method's behavior. The flags can be a combination of the following values: 1 The FindString method wraps to the beginning of the screen or field after reaching the end. 2 The FindString method searches backward.	Integer

Continued next page...

Parameters (*continued*)

Name	Description	Type
inFlags *(continued)*	4 The FindString method searches only in the field that contains the specified row and column. All other flags apply only within the field. The search begins at the point in the field specified by the inRow and inColumn parameters. The outRow and outColumn parameters contain the position on the screen where the string is found, or zero if the string is not found. 8 The FindString method ignores case when searching for a string. **Note:** If you send a zero in the inFlags parameter, this method performs a screen-wide, case-sensitive, non-wrapping, forward search.	
outRow	The row position where the string was found. If the string was not found, this returns a zero.	Integer
outColumn	The column position where the string was found. If the string was not found, this returns a zero.	Integer

GetAllFieldInformation

This method retrieves the information of all fields on the current screen.

Syntax

```
GetAllFieldInformation inScreen, outFieldCount, outFieldNames,
outFieldValues, outFieldAttributes, outFieldRows, outFieldCols,
outFieldLengths
```

The parameter *outFieldCount* is returned as an Integer; *outFieldNames* and *outFieldValues* are String arrays, and all other Arrays are of type Integer. All Arrays occur *outFieldCount* times.

For example, when *outFieldCount* is 10, *outFieldNames* contains an array consisting of 10 Strings, with each String containing a field name. Likewise, *outFieldAttributes* contains an array consisting of 10 Integers, with each containing a value for the corresponding field attribute.

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
outFieldCount	Total number of fields on the current screen. Each array returned by this API contains this number of entries.	Integer
outFieldNames	Names of 3270 fields on the current screen.	String Array
outFieldValues	Values contained in all the fields on the current screen.	String Array
outFieldAttributes	Field attributes of all the fields on the current screen. Please see the reference for the GetFieldAttributes API in this publication for a list of possible values	Integer Array
outFieldRows	Row coordinates for each of the fields on the current screen.	Integer Array
outFieldColumns	Column coordinates for each of the fields on the current screen.	Integer Array
outFieldLengths	Lengths of each of the fields on the current screen	Integer Array

GetAllFieldInformation (Overloaded Method)

This method retrieves the information of all fields on the current screen, including field color and highlighting. This overloaded method is used to extract extended attribute information about the fields on the screen.

Syntax

```
GetAllFieldInformation inScreen, outFieldCount, outFieldNames,
outFieldValues, outFieldAttributes, outFieldRows, outFieldCols,
outFieldLengths, outFieldColors, outFieldHighlighting
```

The parameter *outFieldCount* is returned as an Integer; *outFieldNames* and *outFieldValues* are String arrays, and all other Arrays are of type Integer. All Arrays occur *outFieldCount* times.

For example, when *outFieldCount* is 10, *outFieldNames* contains an array consisting of 10 Strings, with each String containing a field name. Likewise, *outFieldAttributes* contains an array consisting of 10 Integers, with each containing a value for the corresponding field attribute.

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
outFieldCount	Total number of fields on the current screen. Each array returned by this API contains this number of entries.	Integer
outFieldNames	Names of 3270 fields on the current screen.	String Array
outFieldValues	Values contained in all the fields on the current screen.	String Array
outFieldAttributes	Field attributes of all the fields on the current screen. Please see the reference for the GetFieldAttributes API in this publication for a list of possible values.	Integer Array
outFieldRows	Row coordinates for each of the fields on the current screen.	Integer Array

Continued next page...

Parameters (*continued*)

outFieldColumns	Column coordinates for each of the fields on the current screen.	Integer Array
outFieldLengths	Lengths of each of the fields on the current screen.	Integer Array
outFieldColors	Colors in use in each of the fields on the current screen. Each Integer returned contains a value that represents the color: 0 White 1 Blue 2 Red 3 Pink 4 Green 5 Turquoise 6 Yellow 7 White	Integer Array
outFieldHighlights	Field highlighting in use in the fields on the current screen. Each Integer returned contains a value that represents the highlighting currently displayed for the respective 3270 field. The integer representation is as follows: 0 Normal 1 Blinking 2 Reversed video 3 Underscore	Integer Array

GetConnectionStatus

This method returns the status of the host connection.

Syntax

```
GetConnectionStatus outStatus
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
outStatus	The status of the host connection, which is one of the following values: 0 The session is connected. 1 The session is not connected. The following values are not supported: 2 The session is connected but the presentation space (PS) is busy. 3 The connection is inhibited. 4 The host is in an unknown state.	Integer

GetCurrentScreenName

This method returns the name of the current host screen.

Syntax

```
GetCurrentScreenName inTimeOut, outScreenName
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inTimeOut	In this implementation, this parameter is disregarded. It is included only for backward compatibility.	Integer
outScreenName	The name of the current screen.	String

GetCursorPosition

This method retrieves the cursor position on the current host screen.

Syntax

```
GetCursorPosition outRow, outColumn
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
outRow	The cursor row position.	Integer
outColumn	The cursor column position.	Integer

GetFieldAttributes

This method retrieves the attributes of a specified field on the current screen.

Syntax

```
GetFieldAttributes inScreen, inField, outAttributes
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
inField	The field for which to get attributes. This value can be given either as a field name or as the index number of the field on the screen. The index is zero-based.	Integer or String
outFieldNames	The field names of all the fields on the current screen. Returns an array of Strings, with each String containing a field name.	String Array
outAttributes	The returned attributes of the specified field. The attributes can be a combination (sum) of the following: 1 The field is a protected field. 2 The field can contain only numeric values. 4 The field is light-pen detectable. 8 The field is a non-display, non lightpen detectable field. 16 The field is a high-intensity field. The returned value may be the sum of two or more attributes. For example, if 5 is returned, the field is both protected (1) and light-pen detectable (4).	Integer

GetFieldCoordinates

This method retrieves the coordinates (row, column, and length) of a specified field on the current screen.

Syntax

```
GetFieldCoordinates inScreen, inField, outFieldRow, outFieldColumn,
outFieldLength
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
inField	The field for which to get coordinates. This value can be given either as a field name or as the index number of the field on the screen. The index is zero-based.	Integer or String
outFieldRow	The returned row coordinate for the specified field.	Integer
outFieldColumn	The returned column coordinate for the specified field.	Integer
outFieldLength	The length of the specified field.	Integer

GetFieldCount

This method retrieves the number of fields for the current screen.

Syntax

```
GetFieldCount inScreen, outFieldCount
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
outFieldCount	The returned number of fields for the given screen.	Integer

GetFieldNameFromIndex

This method retrieves the field name based on its index on the current screen. The index numbers start at zero and end at 1 less than the field count.

For example, the third field on the screen has a field index of 2. Use GetAllFieldInformation to determine the index values for all fields on a given screen.

Syntax

```
GetFieldNameFromIndex inScreen, inFieldIndex, outFieldName
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or String
inFieldIndex	The index number of the field on the screen. The index is zero-based. This number must be less than the number of fields returned in GetFieldCount.	Integer
outFieldName	The returned field name.	String

GetFieldValue

This method retrieves the value of a specified field on the current screen.

Caution: The GetFieldValue method returns the correct value even for values contained in non-display fields. Therefore, be sure to check the field attribute before displaying text retrieved with this method.

Syntax

```
GetFieldValue inScreen, inField, outValue
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or string
inField	The field for which you are requesting a value. This value can be given either as a field name or as the index number of the field on the screen. The index is zero-based.	Integer or string
outValue	The returned value of the specified field.	String

GetScreenInformation

This method retrieves all available information for the current screen.

Syntax

```
GetScreenInformation outCursorPosition, outAidKey ,outFieldCount,
outScreenName, outFieldNames, outFieldValues,
out3270FieldAttributes, outFieldRows, outFieldCols, outFieldLengths,
outFieldColors, outFieldHighlights
```

The parameters *outCursorPosition* and *outFieldCount* is returned as an Integers; *outAidKey* and *outScreenName* are Strings; *outFieldNames* and *outFieldValues* are String arrays, and all other Arrays are of type Integer. All Arrays occur *outFieldCount* times.

For example, when *outFieldCount* is 10, *outFieldNames* contains an array consisting of 10 Strings, with each String containing a field name. Likewise, *outFieldAttributes* contains an array consisting of 10 Integers, with each containing a value for the corresponding field attribute.

Parameters

Name	Description	Type
outCursor	Offset (relative to zero) of the current logical cursor position.	Integer
outAidkey	String representing the HLLAPI style AID key last processed by the host.. Please see the AID key values in the PC to Host Key Translation table in this publication.	String
outFieldCount	Number of 3270 fields represented on the current logical 3270 screen.	Integer
outScreenName	Name of the 3270 screen represented on the current screen. For BMS names this is the concatenation of the BMS Mapset name, a period, and the BMS Mapname For example "XTR1M40.XTR1A002" If the CICS application that created the display did not use BMS mapping the name "NOMAPSET.NOFIELD-" is returned.	String
outFieldNames	The field names for all the fields on the current screen.	String Array
outFieldValues	The field values of all the fields on the current screen.	String Array

Continued on next page…

Parameters (continued)

Name	Description	Type
out3270FA	The 3270 Field Attributes of all the fields on the current screen. Returns an array with each position containing a real 3270 FA. Bit: 2 0 = Unprotected 1 = Protected 3 0 = Alphanumeric 1 = Numeric * Note '11' in bits 2 and 3 * denotes Autoskip field 4-5 00 = Normal nonlightpen 01 = Normal/lightpen 10 = Bright/lightpen 11 = non-display 6 reserved 7 0 = Not modified 1 = Modified The InterpretFieldAttribute API may be used to translate this 3270 FA byte into an HPS style attribute as defined in the GetFieldAttributes section of this publication.	Byte Array
outFieldRows	Row coordinates for all the fields on the current screen.	Integer Array
outFieldColumns	Column coordinates for all the fields on the current screen.	Integer Array
outFieldLengths	Lengths of all the fields on the current screen.	Integer Array
outFieldColors	Color representations for all the fields on the current screen. Please see GetAllFieldInformation for a definition of color values.	Integer Array
outFieldHighlights	Highlighting representation for all the fields on the current screen. Please see GetAllFieldInformation for a definition of highlighting values.	Integer Array

GetScreenSize

This method retrieves the size of the current screen.

Syntax

GetScreenSize *inScreen, outRows, outColumns*

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or string
outRows	The returned number of rows for the current screen.	Integer
outColumns	The returned number of columns for the current screen.	Integer

GetString

This method retrieves any text on the current screen at the specified row, column, and length. If the length parameter is zero, GetString retrieves text from the specified row and column to the end of the screen. The Row and Column parameters must always be greater than or equal to 1.

Caution: The GetString method returns the correct value even for values contained in non-display fields. Therefore, be sure to check the field attribute before displaying text retrieved with this method.

Syntax

```
GetString inScreen, inRow, inColumn, inLength, outText
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inScreen	The current live host screen. If this value is given as the index number of the screen, it must be –1; if it is given as a string, it must be empty or spaces.	Integer or string
inRow	The row position on the host screen at which to begin retrieving text.	Integer
inColumn	The column position on the host screen at which to begin retrieving text.	Integer
inLength	The length of the text string to be retrieved. If this value is zero, the text is retrieved up to the end of the screen.	Integer
outText	The text that is retrieved.	String

GetTransactionSystemId

This method supplies information as to the execution location of the requested transaction ID. If the requested transaction ID executes locally in the specified CICS region, this method returns an outSystemIdentification value of spaces. If the requested transaction ID is defined in the specified region as residing in another region, this method returns the value of that other region's system ID from the REMOTESystem specification of the transaction table. If the transaction ID is unknown to this region, both locally and through its transaction table's REMOTESystem information, a method return code of 718 is returned.

Syntax

```
GetTransactionSysemId inHostIP, inHostPort, inTransactionId,
outSystemIdentification
```

Prerequisites

None

Parameters

Name	Description	Type
inHostIP	The IP address of the CICS region to be interrogated for the specified transaction.	String
InHostPort	The TCP port of the CICS region interrogated for the specified transaction.	Integer
inTransactionID	A one to four character string that identifies the transaction to inquire.	String
outSystemId	A four character string that identifies the system ID that executes the specified transaction.	String

Initialize

Not required. Executes as a NOP. Included for backward compatibility only.

Syntax

```
Initialize inSessionTermType
```

Prerequisites

None

Parameter

Name	Description	Type
inSessionTermType	In this implementation, this parameter is disregarded. It is included only for backward compatibility.	Integer

InterpretFieldAttribute

Convert a real 3270 Field Attribute value into an HPS style numeric attribute.

Syntax

```
InterpretFieldAttribute in3270FA
```

Prerequisites

None

Parameter

Name	Description	Type
In3270FA	Byte value of a real 3270 FA that is to be translated into an HPS style numeric attribute.	Unsigned Integer

Returns

Integer value containing an HPS-style representation of the attribute byte as follows:

Description	Type
The returned attributes of the specified field. The attributes can be a combination (sum) of the following: 1 The field is a protected field. 2 The field can contain only numeric values. 4 The field is light-pen detectable. 8 The field is a non-display, non lightpen detectable field. 16 The field is a high-intensity field. The returned value may be the sum of two or more attributes. For example, if 5 is returned, the field is both protected (1) and light-pen detectable (4).	Integer

Pause

This method causes execution to pause for the amount of time that is indicated in the inTime parameter. This method does not return until the pause time has elapsed.

Syntax

```
Pause inTime
```

Prerequisites

None

Parameters

Name	Description	Type
inTime	The amount of time to pause, in milliseconds.	Integer

PutString

This method writes text to the session presentation space beginning at the row and column specified. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

Syntax

```
PutString inText, inRow, inColumn, outNumChars
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inText	The text string to write, beginning at the specified position.	String
inRow	The row position on the host screen at which to begin writing the text string.	Integer
inColumn	The column position on the host screen at which to begin writing the text string.	Integer
outNumChars	The returned number of characters that were actually written to the screen.	Integer

SendKeys

This method writes text to the session presentation space beginning at the current cursor position. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

If the HLLAPI representation of an AID keystroke is encountered in the text string, SendKeys initiates transmission to the host of the AID key followed by any modified fields in the session presentation space. Text and AID keystrokes can be combined multiple times in the input string in this implementation.

Syntax

```
SendKeys inKeys
```

Prerequisites

ConnectBridge

Parameter

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used.	String

PC to Host Key Translation

PC to host key translation enables the use of ASCII characters to represent the special functions of a host keyboard. Host AID keys are identified in the Comments column of the following table. Also referred to as Action Keys or Workstation Control keys (WsCtrl), these are the keys on a terminal keyboard that send host functions to the mainframe.

Note: The key combinations are case-sensitive. For example, "@A" refers to a different host key than "@a." Each PC key code represents the actual keystroke that is sent.

Host Key	PC Keys	Comments
Backtab	@B	
Clear	@C	Host AID key
Enter	@E	Host AID key
Erase EOF	@F	
Cursor Left	@L	
New Line (line feed)	@N	
Tab	@T	
Cursor Up	@U	
Cursor Down	@V	
Cursor Right	@Z	
Backspace	@<	
Home	@0	
PF1	@1	Host AID key
PF2	@2	Host AID key
PF3	@3	Host AID key
PF4	@4	Host AID key
PF5	@5	Host AID key
PF6	@6	Host AID key
PF7	@7	Host AID key
PF8	@8	Host AID key
PF9	@9	Host AID key
PF10	@a	Host AID key
PF11	@b	Host AID key

PF12	@c	Host AID key
PF13	@d	Host AID key
PF14	@e	Host AID key
PF15	@f	Host AID key
PF16	@g	Host AID key
PF17	@h	Host AID key
PF18	@i	Host AID key
PF19	@j	Host AID key
PF20	@k	Host AID key
PF21	@l	Host AID key
PF22	@m	Host AID key
PF23	@n	Host AID key
PF24	@o	Host AID key
PA1	@x	Host AID key
PA2	@y	Host AID key
PA3	@z	Host AID key
ErInp	@A@F	

SendKeysEx

This method is included for backward compatibility only. It is implemented as a SendKeys method call.

Syntax

```
SendKeys inKeys
```

Prerequisites

ConnectBridge

Parameter

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used.	String

SendKeysGetAllFieldInformation

This method combines a SendKeys and a GetAllFieldInformation in a single call to the host.

The SendKeys part of this method writes text to the session presentation space beginning at the current cursor position. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

If the HLLAPI representation of an AID keystroke is encountered in the text string, SendKeys initiates transmission to the host of the AID key followed by any modified fields in the session presentation space. Text and AID keystrokes can be combined multiple times in the input string in this implementation.

The GetAllFieldInformation part of this method retrieves the information of all fields on the current screen.

Syntax

```
SendKeysGetAllFieldInformation inKey, outFieldCount, outFieldNames,
outFieldValues, outFieldAttributes, outFieldRows, outFieldCols,
outFieldLengths, outFieldColors, outFieldHighlights
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used.	String
outFieldCount	The total number of fields on the current screen.	Long Integer
outFieldNames	The field names of all the fields on the current screen. Returns an array of Strings, with each String containing a field name.	String Array
outFieldValues	The field values contained in all the field on the current screen. Returns an array of Strings with each String containing the current field value for each field.	String Array
outFieldAttributes	The field attributes of all the fields on the current screen. Returns an array of Long Integers, with each Long Integer containing a LONG for the field attribute.	Long Integer Array
outFieldRows	The coordinates of rows of all the fields on the current screen. Returns an array of Long Integers, with each Long Integer containing a LONG for the row coordinate.	Long Integer Array
outFieldCols	The coordinates of columns of all the fields on the current screen. Returns an array of Long Integers, with each Long Integer containing a LONG for the column coordinate.	Long Integer Array

Continued on next page…

Parameters (*continued*)

outFieldLengths	The lengths of all the fields on the current screen. Returns an array of Long Integers, with each Long Integer containing a LONG for the field length.	Long Integer Array
outFieldColors	The field colors in use in the fields on the current screen. Returns an array of Long Integers, with each Long Integer containing a LONG for the value that represents the color: 0 White 1 Blue 2 Red 3 Pink 4 Green 5 Turquoise 6 Yellow 7 White	Long Integer Array
outFieldHighlights	The field highlighting in use in the fields on the current screen. Returns an array of Long Integers, with each Long Integer containing a LONG for the value that represents the highlighting currently displayed for the respective 3270 field: 0 Normal 1 Blinking 2 Reversed video 3 Underscore	Long Integer Array

SendKeysGetString

This method combines a SendKeys and a GetString into a single call to the host.

The SendKeys part of this method writes text to the session presentation space beginning at the current cursor position. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

If the HLLAPI representation of an AID keystroke is encountered in the text string, SendKeys initiates transmission to the host of the AID key followed by any modified fields in the session presentation space. Text and AID keystrokes can be combined multiple times in the input string in this implementation.

The GetString part of this method retrieves any text on the current screen at the specified row, column, and length. If the length parameter is zero, GetString retrieves text from the specified row and column to the end of the screen. The Row and Column parameters must always be greater than or equal to 1.

Caution: The SendKeysGetString method returns the correct value, even though it is contained in non-display fields. Therefore, be sure to check the field attribute prior to displaying text retrieved with this method.

Syntax

```
SetCursorPosition inRow, inColumn
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used.	String
inRow	The row position on the host screen at which to begin retrieving text.	Integer
inColumn	The column position on the host screen at which to begin retrieving text.	Integer
inLength	The length of the text string to be retrieved. If this value is zero, the text is retrieved up to the end of the screen.	Integer
outText	The text that is retrieved.	String

SetCursorPosition

This method sets the cursor position on the current host screen.

Syntax

```
SetCursorPosition inRow, inColumn
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inRow	The cursor row position. The first row is row 1.	Integer
inColumn	The cursor column position. The first column is column 1.	Integer

SetFieldValue

This method sets the value of a specified unprotected field on the current screen

Syntax

```
SetFieldValue inField, inValue
```

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
inField	The field for which a value is to be set. This value can be given either as a field name or as the index number of the field on the screen. The index is zero-based.	Integer or String
inValue	The value to set for the specified field.	String

SetFieldValue (Overloaded Method)

This method sets the value of an array of specified unprotected fields on the current screen with a single call. Use of this method will increase performance of applications when updating multiple fields.

Syntax

SetFieldValue *infields(), inValues()*

Prerequisites

ConnectBridge

Parameters

Name	Description	Type
infields	An array of type String, each occurrence of which describes the name of the field to have its value replaced.	String
inValues	An array of type String, each occurrence of which specifies the new value to be placed into the above-named field.	String

Example

```
Dim setFieldNames(4) As String   Dim setFieldValues(4) As String

'Fill out the bulk fruit order screen

setFieldNames(0) = "FRUIT"

setFieldValues(0) = "APPLES"

setFieldNames(1) = "QNTY"

setFieldValues(1) = "1 BOX"

setFieldNames(2) = "TYPE"

setFieldValues(2) = "GOLDEN DELICIOUS"

setFieldNames(3) = "CUSTNM"

setFieldValues(3) = "BOBS APPLE STAND"

rc = SetFieldValue(setFieldNames, setFieldValues)
```

SetSettleTime

This method is included for backward compatibility only. It is implemented as a NOP. There is no settle time requirement with the host data engine.

Syntax

SetSettleTime *inSettleTime*

Prerequisites

None

Parameter

Name	Description	Type
inSettleTime	In this implementation, this parameter is disregarded. It is included only for backward compatibility.	Integer

SetTimeOut

This method is included for backward compatibility only. It is implemented as a NOP.

Syntax

```
SetTimeOut inTimeOut
```

Prerequisites

None

Parameter

Name	Description	Type
inTimeOut	In this implementation, this parameter is disregarded. It is included only for backward compatibility.	Integer

SetTraceActive

This method sets the connector's trace facility active for this session, and optionally identifies the specific CICS Temporary Storage Queue for holding the trace output. If the queue name is supplied, but the queue does not exist in the CICS region handling this session, it will be created as a result of this request. Inactive is the default starting status for the trace facility.

Note: CICS Temporary Storage Queues are not deleted from this API. They are automatically removed when the CICS region is restarted. They can be removed at any time using the proper and authorized CICS commands.

Syntax

```
SetTraceActive TraceQueueName
```

Prerequisites

ConnectBridge

Parameter

Name	Description	Type
TraceQueueName	The name of the CICS Temporary Storage Queue to be used for trace output.	String

SetTraceActiveInactive

This method sets the connector's trace facility inactive for this session. Inactive is the default starting status for the trace facility.

Syntax

```
SetTraceInactive
```

Prerequisites

ConnectBridge

SetTraceName

This method identifies the specific CICS Temporary Storage Queue for holding the trace output. If the queue name is supplied, but the queue does not exist in the CICS region handling this session, it is created as a result of this request. If a different queue name was previously associated with the trace facility for this session, it will be changed to the supplied trace queue name in this method call.

Note: CICS Temporary Storage Queues are not deleted from this API. They are automatically removed when the CICS region is restarted. They can be removed at any time using the proper and authorized CICS commands.

Syntax

```
SetTraceName TraceQueueName
```

Prerequisites

ConnectBridge

Parameter

Name	Description	Type
TraceQueueName	The name of the CICS Temporary Storage Queue to be used for trace output.	String

Terminate

This method is included for backward compatibility only. It is implemented as a NOP.

Syntax

```
Terminate
```

Prerequisites

None

X3270Screen

This method combines a SendKeys and a SetFieldValue with GetScreenInformation in a single call to the host.

SendKeys

The SendKeys part of this method writes text to the session presentation space beginning at the current cursor position. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

If the HLLAPI representation of an AID keystroke is encountered in the text string, SendKeys initiates transmission to the host of the AID key followed by any modified fields in the session presentation space. Text and AID keystrokes can be combined multiple times in the input string in this implementation.

SetFieldValue

The SetFieldValue part of this method allows the placing of field-index/value paired data into the indentified field in the screen buffer when the Mnemonic "@A@P" is encountered within the SendKeys string.

GetScreenInformation

The GetScreenInformation part of this method retrieves all information from the 3270 screen (from either the end of the Sendkeys string or as remembered when encountering the "@A@G" Mnemonic while processing the SendKeys string).

Optionally, the special mnemonic "@A@X" can be sent to indicate that after the GetScreenInformation is returned to the client the host will automatically Disconnect() the session on the client's behalf thus saving the need to initiate a Disconnect() API exchange with the host and dispatch the related network traffic.


This method is a multi-operation function that is intended for use only by advanced VBI developers

Parameters

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used along with 3 additional mnemonics: @A@P Place SetFieldValue data @A@G GetScreenInformation data @A@X Disconnect the VBI session	String
inFieldIndex	3270 Field index in which the data in inFieldValue is to be placed. Please see the SetFieldValue API in this publication.	Integer
inFieldValues	Data to place into the 3270 entry field as part of the SetFieldValue.	String
outCursor	Offset (relative to zero) of the current logical cursor position.	Integer
outAidkey	String representing the HLLAPI style AID key last processed by the host.. Please see the AID key values in the PC to Host Key Translation table in this publication.	String
outFieldCount	Number of 3270 fields represented on the current logical 3270 screen.	Integer
outScreenName	Name of the 3270 screen represented on the current screen. For BMS names this is the concatenation of the BMS Mapset name, a period, and the BMS Mapname For example "XTR1M40.XTR1A002" If the CICS application that created the display did not use BMS mapping the name "NOMAPSET.NOFIELD-" is returned.	String
outFieldNames	The field names for all the fields on the current screen.	String Array
outFieldValues	The field values of all the fields on the current screen.	String Array

Continued on next page…

Parameters (continued)

Name	Description	Type
out3270FA	The 3270 Field Attributes of all the fields on the current screen. Returns an array with each position containing a real 3270 FA. Bit: 2 0 = Unprotected 1 = Protected 3 0 = Alphanumeric 1 = Numeric * Note '11' in bits 2 and 3 * denotes Autoskip field 4-5 00 = Normal nonlightpen 01 = Normal/lightpen 10 = Bright/lightpen 11 = non-display 6 reserved 7 0 = Not modified 1 = Modified The InterpretFieldAttribute API may be used to translate this 3270 FA byte into an HPS style attribute as defined in the GetFieldAttributes section of this publication.	Byte Array
outFieldRows	Row coordinates for all the fields on the current screen.	Integer Array
outFieldColumns	Column coordinates for all the fields on the current screen.	Integer Array
outFieldLengths	Lengths of all the fields on the current screen.	Integer Array
outFieldColors	Color representations for all the fields on the current screen. Please see GetAllFieldInformation for a definition of color values.	Integer Array
outFieldHighlights	Highlighting representation for all the fields on the current screen. Please see GetAllFieldInformation for a definition of highlighting values.	Integer Array

X3270Screen (Overloaded Method)

This method combines a SendKeys and a SetFieldValue with GetScreenInformation in a single call to the host.

SendKeys

The SendKeys part of this method writes text to the session presentation space beginning at the current cursor position. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

If the HLLAPI representation of an AID keystroke is encountered in the text string, SendKeys initiates transmission to the host of the AID key followed by any modified fields in the session presentation space. Text and AID keystrokes can be combined multiple times in the input string in this implementation.

SetFieldValue

The SetFieldValue part of this method allows the placing of name/value paired data into the identified field in the screen buffer when the mnemonic "@A@P" is encountered within the SendKeys string.

GetScreenInformation

The GetScreenInformation part of this method retrieves all information from the 3270 screen (from either the end of the Sendkeys string or as remembered when encountering the "@A@G" Mnemonic while processing the SendKeys string).

Optionally, the special mnemonic "@A@X" can be sent to indicate that after the GetScreenInformation is returned to the client the host will automatically Disconnect() the session on the client's behalf thus saving the need to initiate a Disconnect() API exchange with the host and dispatch the related network traffic.


This method is a multi-operation function that is intended for use only by advanced VBI developers

Parameters

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used along with 3 additional mnemonics: @A@P Place SetFieldValue data @A@G GetScreenInformation data @A@X Disconnect the VBI session	String
inFieldName	3270 Field name in which the data in inFieldValue is to be placed. Please see the SetFieldValue API in this publication.	String
inFieldValue	Data to place into the indentified 3270 entry field. Please see the SetFieldValue API in this publication.	String
outCursor	Offset (relative to zero) of the current logical cursor position.	Integer
outAidkey	String representing the HLLAPI style AID key last processed by the host.. Please see the AID key values in the PC to Host Key Translation table in this publication.	String
outFieldCount	Number of 3270 fields represented on the current logical 3270 screen.	Integer
outScreenName	Name of the 3270 screen represented on the current screen. For BMS names this is the concatenation of the BMS Mapset name, a period, and the BMS Mapname For example "XTR1M40.XTR1A002" If the CICS application that created the display did not use BMS mapping the name "NOMAPSET.NOFIELD-" is returned.	String
outFieldNames	The field names for all the fields on the current screen.	String Array
outFieldValues	The field values of all the fields on the current screen.	String Array

Continued on next page…

Parameters (continued)

Name	Description	Type
out3270FA	The 3270 Field Attributes of all the fields on the current screen. Returns an array with each position containing a real 3270 FA. Bit: 2 0 = Unprotected 1 = Protected 3 0 = Alphanumeric 1 = Numeric * Note '11' in bits 2 and 3 * denotes Autoskip field 4-5 00 = Normal nonlightpen 01 = Normal/lightpen 10 = Bright/lightpen 11 = non-display 6 reserved 7 0 = Not modified 1 = Modified The InterpretFieldAttribute API may be used to translate this 3270 FA byte into an HPS style attribute as defined in the GetFieldAttributes section of this publication.	Byte Array
outFieldRows	Row coordinates for all the fields on the current screen.	Integer Array
outFieldColumns	Column coordinates for all the fields on the current screen.	Integer Array
outFieldLengths	Lengths of all the fields on the current screen.	Integer Array
outFieldColors	Color representations for all the fields on the current screen. Please see GetAllFieldInformation for a definition of color values.	Integer Array
outFieldHighlights	Highlighting representation for all the fields on the current screen. Please see GetAllFieldInformation for a definition of highlighting values.	Integer Array

X3270Screen (Overloaded Method)

This method combines a SendKeys and a SetFieldValue with GetScreenInformation in a single call to the host.

SendKeys

The SendKeys part of this method writes text to the session presentation space beginning at the current cursor position. If a protected field is encountered while writing, characters from the text string are discarded, and writing continues with the next unprotected field.

If the HLLAPI representation of an AID keystroke is encountered in the text string, SendKeys initiates transmission to the host of the AID key followed by any modified fields in the session presentation space. Text and AID keystrokes can be combined multiple times in the input string in this implementation.

SetFieldValue

The SetFieldValue part of this method allows the placing of a coordinated array set of name/value paired data into the indentified fields in the screen buffer when the Mnemonic "@A@P" is encountered within the SendKeys string.

GetScreenInformation

The GetScreenInformation part of this method retrieves all information from the 3270 screen (from either the end of the Sendkeys string or as remembered when encountering the "@A@G" Mnemonic while processing the SendKeys string).

Optionally, the special mnemonic "@A@X" can be sent to indicate that after the GetScreenInformation is returned to the client the host will automatically Disconnect() the session on the client's behalf thus saving the need to initiate a Disconnect() API exchange with the host and dispatch the related network traffic.


This method is a multi-operation function that is intended for use only by advanced VBI developers

Parameters

Name	Description	Type
inKeys	The keys to send to the host. The HLLAPI representations for special keys can be used along with 3 additional mnemonics: @A@P Place SetFieldValue data @A@G GetScreenInformation data @A@X Disconnect the VBI session	String
inFieldName	3270 Field names in which the data in inFieldValue is to be placed. See the SetFieldValue API	String Array
inFieldValue	Data to place into the indentified 3270 entry field. See the SetFieldValue API.	String Array
outCursor	Offset (relative to zero) of the current logical cursor position.	Integer
outAidkey	String representing the HLLAPI style AID key last processed by the host. Please see the AID key values in the PC to Host Key Translation table in this publication.	String
outFieldCount	Number of 3270 fields represented on the current logical 3270 screen.	Integer
outScreenName	Name of the 3270 screen represented on the current screen. For BMS names this is the concatenation of the BMS Mapset name, a period, and the BMS Mapname For example "XTR1M40.XTR1A002" If the CICS application that created the display did not use BMS mapping the name "NOMAPSET.NOFIELD-" is returned.	String
outFieldNames	The field names for all the fields on the current screen.	String Array
outFieldValues	The field values of all the fields on the current screen.	String Array

Continued on next page…

Parameters (*continued*)

Name	Description	Type
out3270FA	The 3270 Field Attributes of all the fields on the current screen. Returns an array with each position containing a real 3270 FA. Bit: 2 0 = Unprotected 1 = Protected 3 0 = Alphanumeric 1 = Numeric * Note '11' in bits 2 and 3 * denotes Autoskip field 4-5 00 = Normal nonlightpen 01 = Normal/lightpen 10 = Bright/lightpen 11 = non-display 6 reserved 7 0 = Not modified 1 = Modified The InterpretFieldAttribute API may be used to translate this 3270 FA byte into an HPS style attribute as defined in the GetFieldAttributes section of this publication.	Byte Array
outFieldRows	Row coordinates for all the fields on the current screen.	Integer Array
outFieldColumns	Column coordinates for all the fields on the current screen.	Integer Array
outFieldLengths	Lengths of all the fields on the current screen.	Integer Array
outFieldColors	Color representations for all the fields on the current screen. Please see GetAllFieldInformation for a definition of color values.	Integer Array
outFieldHighlights	Highlighting representation for all the fields on the current screen. Please see GetAllFieldInformation for a definition of highlighting values.	Integer Array

Example

An example of a 3270 application that requires:

- Connection of the VBI client to the CICS region

- Transaction "ABCD"
- Menu selection of entering the character '1' into a menu choice 3270 field
- An <enter> 3270 AID key
- Entry of a client account number "12345" for display of a specific account
- Another <enter> 3270 AID key
- Collection of all information from the 3270 screen returned by the CICS application
- A <PF3> 3270 AID key to exit out of this account display screen
- A <clear> 3270 AID key to exit completely out of the CICS application
- A Disconnect() VBI API to disconnect from the session.

One could accomplish all of this with two trips to the mainframe CICS region as follows:

```
…
                Dim RC As Integer
Dim outCursorPosition As Integer
Dim outAidKey As String
Dim outFieldCount As Integer
Dim outScreenName As String
Dim outFieldNames() As String
Dim outFieldValues() As String
Dim out3270FieldAttributes() As Byte
Dim outFieldRows() As Integer
Dim outFieldCols() As Integer
Dim outFieldLengths() As Integer
Dim outFieldColors() As Integer
Dim outFieldHighlights() As Integer

                RC = Bridge.ConnectBridge("140.147.249.7", 2317, 2,
                                          "MYUSERID", "MYPASSWD")

RC = Bridge.X3270Screen( "ABCD@E@T1@E@A@P@E@A@G@3@3@A@X",
                                                  “ACCOUNT”,
                                                  "12345",
                                                  outCursorPosition,
                                                  outAidKey,
                                                  outFieldCount,
                                                  outScreenName,
                                                  outFieldNames,
                                                  outFieldValues,
                                                  out3270FieldAttributes,
                                                  outFieldRows,
                                                  outFieldCols,
                                                  outFieldLengths,
                                                  outFieldColors,
                                                  outFieldHighlights)
…
```

Error Codes

Mainframe Server Error Codes

Return Code	Description
200	CICS COMMAREA HAD A LENGTH OF ZERO AT THE START
201	UNRECOGNIZED REQUEST FUNCTION CODE IN COMMAREA
202	UNRECOGNIZED EYECATCHER IN REQUEST COMMAREA
210	INQUIRE TERMINAL FOR CASE FAILED FOR CONNECT
212	INQUIRE TERMINAL FOR CONVERSE FAILED
214	QUERY TERMINAL FAILED
215	TERMINAL ID IN USE
320	TERMINAL MODEL NUMBER IN CONNECT REQUEST WAS INVALID
330	BUFFER SIZE FROM SESSION STATE IS NOT A VALID BUFFSIZE
340	USER ID/PASSWORD COMBINATION INVALID
350	INVALID POOL INFORMATION
351	RESOURCE ALREADY EXISTS
352	TERMINAL POOL TEMPORARY STORAGE IO ERROR
353	TERMINAL RESOURCE IN POOL INACTIVE
354	NO TERMINAL POOL CONFIGURATION
355	POOL TERMINAL UNAVAILABLE
401	CICS GETMAIN FOR MSG Q STAGING AREA FAILED
410	CICS GETMAIN FOR REPLY VECTOR FAILED
460	CICS GETMAIN FOR CREATION OF SESSION STATE DATA FAILED
461	CICS GETMAIN FOR CREATION OF MSG AND REPLY ECBS FAILED

Continued on next page…

Mainframe Server Error Codes (*continued*)

Return Code	Description
470	CICS GETMAIN TO HOLD SESSION STATE DATA FAILED
480	FAILED TO OBTAIN VIRTUAL TERMINAL (MRO)
490	TERMINAL FACILITY RECREATE FAILED (MRO)
600	CICS WRITE TO THE BRIDGE MSG TSQ RESOURCE FAILED
610	CICS WRITE TO THE SESSION STATE Q RESOURCE FAILED
620	CICS READ OF THE BRIDGE REPLY TSQ RESOURCE FAILED
649	CICS READQ TS FAILED FOR TERMINAL CHECK-IN
650	CICS READ OF THE STATE TSQ RESOURCE FAILED
651	REQUESTED SESSION IS NOT ACTIVE OR VALID
700	CICS START APPLICATION TRAN(xxxx) WITH BRIDGE FAILED
710	TIMEOUT ON A READ OF THE BRIDGE REPLYQ
715	BRIDGE EXIT DETECTED FAILURE WHILE PROCESSING TRAN = xxxx AB=yyyy
718	TRANSACTION INQUIRY FAILED
750	TRACE REQUEST FLAG CONTAINS INVALID VALUE
755	TRACE REQUEST QNAME IS INVALID
760	HOST APPLICATION ISSUED AN EXPLICIT ROLLBACK
800	UNRECOG BRIDGE REQUEST VECTOR RETURNED IN REPLY Q DATA
801	UNHANDLED BRIDGE VECTOR: VECTOR = xxxx
810	UNRECOG METHOD CODE FOR GET SECURITY INFO IN CONNECT

Continued on next page…

Mainframe Server Error Codes (*continued*)

Return Code	Description
999	UNEXPECTED OR UNKNOWN ERROR CONDITION
1001	CICS START OF CLEANUP TRAN(xxxx) FAILED
1002	CICS CANCEL OF STARTED ICE FAILED
1003	CICS RETRIEVE OF DATA ASSOC WITH ICE START, FAILED
1010	INVALID FINDSTRING SCREEN,ROW,COL,STRING COMBINATION
1100	CICS SEND MAP WITH MAPPINGDEV AND FRSET FAILED
1110	CICS SEND MAP WITH MAPPINGDEV NO FRSET FAILED
1120	CICS SEND MAP FOR CESN WITH MAPPINGDEV NO FRSET FAILED
1200	CICS LOAD PROGRAM WITH MAPSET FAILED
1210	CICS RELEASE PROGRAM WITH MAPSET FAILED
1300	SENDKEYS STRING LENGTH = ZERO
1301	INVALID SCREEN INDEX OR SCREEN NAME VALUE
1302	PUTSTRING -> PUTSTRINGEX TO VCUT DISPLAY BUFFER FAILED
1303	PUTSTRINGEX TO VCUT DISPLAY BUFFER FAILED
1305	SENDKEYS STRING INVALID, ENDED WITH A SINGLE ESC CHAR
1306	UNEXPECTED OR UNKNOWN REQUEST FUNCTION

Continued on next page…

Mainframe Server Error Codes (*continued*)

Return Code	Description
1307	INVALID GETSTRING ROW,COL,LEN COMBINATION FOR MODEL
1308	RENDER OF AID KEY IN VCUT DISPLAY BUFFER FAILED
1309	PUTSTRING REQUEST ROW IS INVALID FOR MODEL
1310	PUTSTRING REQUEST COL IS INVALID FOR MODEL
1311	PUTSTRING TEXT LENGTH INVALID FOR MODEL BUFFER SIZE
1312	GET FIELD INFORMATION FAILED
1304	SENDKEYS PUTSTRING TO VCUT DISPLAY BUFFER FAILED
1313	SET FIELD VALUE FAILED
1314	GET ALL FIELD INFORMATION FAILED
1320	FIELDCOUNT FAILED
1321	FINDFIELD FAILED
1322	FINDSTRING FAILED
1323	INVALID ROW, COL, POSITION COMBINATION FOR REQUEST
1326	GETATTRIBUTES FAILED
1327	GETSTRINGEX FAILED FOR PASSWORD
1328	GETSTRINGEX FAILED FOR USERID
1329	NO SAF CONFIGURATION FOUND
1330	NOT AUTHORIZED TO ACCESS RESOURCE (SAF)
1331	AUTHORIZATION CONTROL URM ERROR (SAF)
1332	SERVER CONFIGURATION DATA NOT FOUND
1402	CICS VERIFY: INVALID PASSWORD ESMREASON=numberESMRESP=number

Continued on next page…

Mainframe Server Error Codes (*continued*)

Return Code	Description
1403	CICS VERIFY: NEW PASSWORD IS REQUIRED ESMREASON=numberESMRESP=number
1404	CICS VERIFY: NEW PASSWORD NOT ACCEPTED ESMREASON=numberESMRESP=number
1408	CICS VERIFY: USERID NOT KNOWN TO ESM ESMREASON=numberESMRESP=number
1413	CICS VERIFY: UNKNOWN RET FROM THE ESM ESMREASON=numberESMRESP=number
1418	CICS VERIFY: ESM INTERFACE IS NOT INIT ESMREASON=numberESMRESP=number
1419	CICS VERIFY: THE USERID IS REVOKED ESMREASON=numberESMRESP=number
1422	CICS VERIFY: FAILED DURING SECLABEL ESMREASON=numberESMRESP=number
1429	CICS VERIFY: ESM NOT RESPONDING ESMREASON=numberESMRESP=number
1431	CICS VERIFY: USERID REVOKED/DEF GROUP ESMREASON=numberESMRESP=number
1432	CICS VERIFY: INVALID USERID ESMREASON=numberESMRESP=number
1490	CICS VERIFY: UNKNOWN CICS RESP CODE ESMREASON=numberESMRESP=number
1492	CICS VERIFY: UNKNOWN CICS RESP2 CODE ESMREASON=numberESMRESP=number
1497	USERID NOT SUPPLIED OR SPACES
1498	PASSWORD NOT SUPPLIED OR SPACES
1499	NEW PASSWORD NOT SUPPLIED OR SPACES
91000	CREATE FACILITY FOR SESSION FAILED
92000	LINK TO SESSION FACILITY FAILED
93000	DELETE FACILITY FOR SESSION FAILED

FOR THE ABOVE 3 CATEGORIES, MORE INFORMATION MAY BE AVAILABLE, AND IF SO, IT IS INCLUDED AS THE 3 LOW ORDER POSITIONS OF THE RETURN CODE.

For example, if during regular continued session processing with the bridge, an abend occurred in the backend CICS3270 application, it would be signaled as a value of 160 in the 3 low order positions of the return code. Since this occurred during regular processing with a successfully allocated bridge facility (that is, in the category of LINK TO SESSION FACILITY FAILED) the value returned would be:

92000 + 160 = 92160, presented to the COM requestor as -92160.

The 3 position extended codes follow:

20	TERMID FROM AUTOINSTALL IS INVALID
21	NETNAME FROM AUTOINSTALL IS INVALID
22	AUTOINSTALL URM REJECTED BRIDGE INSTALL REQUEST
23	LINK TO AUTOINSTALL URM FAILED
24	SUPPLIED NETNAME INVALID
25	SUPPLIED TERMID INVALID
26	SUPPLIED FACILITYLIKE INVALID
40	LINK TO THE DYNAMIC ROUTING URM
41	BRIDGE ROUTING REQUEST REJECTED BY THE DYNAMIC ROUTING URM
42	THE TRANSACTION DEFINITION DOES NOT ALLOW IT TO BE ROUTED TO THE REQUESTED REGION.
43	TRANSACTION REQUEST COULD NOT BE ROUTED TO THE REMOTE REGION DUE TO A CONNECTION ERROR.
44	LINK TO AOR FAILED WITH TERMERR
45	REQUEST WAS ROUTED TO A BACK LEVEL CICS THAT DOES NOT SUPPORT LINKABLE BRIDGE
61	INVALID FACILITYTOKEN SUPPLIED IN BRIDGE HEADER INFORMATION
62	ALL BRIDGE FACILITIES ARE ALLOCATED
63	THIS FACILITYTOKEN IS IN USE
64	NOT ENOUGH STORAGE TO RUN REQUEST IN ROUTER REGION OR APPLICATION OWNING REGION
65	FILE DFHBRNSF IS UNAVAILABLE OR HAS NOT BEEN DEFINED
66	THE CICS REGION IS TERMINATING, PROCESSING REQUEST REJECTED
80	TRANSACTION NOT ENABLED TO RUN AT SHUTDOWN
82	TRANSACTION CAN ONLY BE SYSTEM ATTACHED, NOT

	VALID UNDER THE BRIDGE
84	TRANSACTION IS DISABLED
85	TRANSACTION IS NOT FOUND
86	TRANSACTION IS NOT RUNNING ON BRIDGE FACILITY
87	TRANSACTION PROFILE IS NOT FOUND
100	USERID IN SESSION REQUEST DIFFERENT THAN USERID SUPPLIED TO ALLOCATE THE BRIDGE FACILITY
120	THE GET MORE DATA REQUEST FAILED BECAUSE THERE WAS NO MORE DATA
121	RETRIEVE VECTORS NOT SUPPORTED AFTER INITIAL REQUEST
140	INVALID DATA LENGTH SPECIFIED IN BRIDGE HEADER
141	VECTOR IS INVALID
142	ALLOCATE FACILITY REQUEST CONTAINED AN INVALID KEEPTIME OF ZERO
143	MESSAGE CONTAINS NO VECTORS: CONTINUE REQUEST IS INVALID
160	TARGET CICS 3270 APPLICATION ABEND

Bridge API Error Codes

Return Code	Description
-9	Invalid terminal model.
-10	No Content-Length token in mainframe response.
-15	No Content-Length token in mainframe response.
-20	No first MIME-PART-BOUNDARY in mainframe response.
-30	No "HTTP/1.0 200 OK" token found in mainframe response.
-35	API function requesting other than the current screen.
-37	No field name supplied.
-38	SendKeys sent a null or zero length string representing the keystrokes to be passed to the mainframe.
-39	Invalid field index value.
-40	Screen name greater than 255 characters.
-50	UserId invalid.
-60	Password Invalid.
-70	No SetField field data supplied.
-85	Unsuccessful TERMID check out.
-99	Invalid Host IP address.
-100	Invalid Content-Length format in mainframe response.

Socket Error Codes

WinSock Error codes are as documented in Winerror.h and WinSock.h.