

EPI Suite OLE Automation

Overview

This feature allows third-party applications to integrate EPI Suite® functionality using OLE automation interfaces. Interfaces are provided for the GuardTool Importer and GuardCard applications, which allow third-party applications to directly execute some of the functionality of these applications. As well, event interfaces are provided to allow third-party applications to take action based on certain EPI Suite events. From the point of view of EPI Suite, event interfaces are outgoing interfaces and are implemented by client applications.

To gain access to the EPI Suite OLE Automation components in Visual Basic:

Select References from the Project menu, then select EPI Suite and click OK.

GuardCardCtrl Object

The GuardCardCtrl object is used to control and receive events from the GuardCard application. GuardCardCtrl implements the IGuardCardCtrl interface and acts as a source of events for the IGuardCardEvents interface.

IGuardCardCtrl Interface

Provides methods to control some GuardCard application functionality, including:

- Open GuardCard with a specific database, user name and password.
- Select one or more person and/or card records for viewing or editing.
- Print cards.
- Initiate image capture.
- Batch export records.
- Shutdown GuardCard.

Methods

GetRunState

Call this function to determine how GuardCard was started; either programmatically, or by the user. When a GuardCardCtrl automation object is created, it will cause GuardCard to startup if an instance was not already running. Note: There is no Start method in the IGuardCardCtrl interface - creating a GuardCardCtrl object automatically starts the application. This function is provided to allow the client application the option to take control of a user initiated GuardCard session.

Syntax

```
ret = gc.GetRunState()
```

Part	Description
ret	[retval] An EPIRunState constant. A value of EPIRunAutomation indicates GuardCard was started programmatically, a value of EPIRunUser indicates GuardCard was started by the user.
gc	A GuardCardCtrl variable.

GetCurrentUser

Get the user name and security level for the user currently using GuardCard.

Syntax

```
gc.GetCurrentUser(pstUser, plUserLevel)
```

Part	Description
gc	A GuardCardCtrl variable.
pstUser	[out] The User ID of the current GuardCard user.
plUserLevel	[out] A long integer that specifies the user security level of current GuardCard user. One of the following values: 1 = administrator 2 = supervisor 3 = clerk 4 = print clerk 5 = viewer

Login

Login to GuardCard. If login is at less than a Administrator user level, a call to EnableSecurityOverride will be required before certain operations can be performed. (ex. Image capture)

Syntax

```
gc.Login(stUser, stPassword)
```

Part	Description
gc	A GuardCardCtrl variable.
stUser	[in] The User ID.
stPassword	[in] The password.

EnableSecurityOverride

This method is provided to allow automation methods to perform operations the current GuardCard user is not permitted to do. For example, a “Clerk” is not allowed to edit person records in GuardCard without an Administrator override. Calling this function enables all automation methods, regardless of the user level of the current user. If this method is not called, other automation methods will fail if the user doesn't have the appropriate rights. For example, start GuardCard with a “Viewer” level user, and then call PrintCard method. It will fail if EnableSecurityOverride was not previously called. Security override will remain enabled until DisableSecurityOverride is called.

Syntax

```
gc.EnableSecurityOverride(stUser, stPassword)
```

Part	Description
gc	A GuardCardCtrl variable.
stUser	[in] An Administrator-level User ID.
stPassword	[in] An Administrator-level password.

DisableSecurityOverride

Disable security override if it was enabled through EnableSecurityOverride.

Syntax

gc.DisableSecurityOverride

Part	Description
gc	A GuardCardCtrl variable.

OpenDatabase

When GuardCard is started through automation, a database will not be opened on startup as usual. OpenDatabase must be called before performing any operations requiring database access. Pass an empty string in strConnect to connect to the default database. Owner and qualifier are optional, pass empty strings if not required.

Syntax

gc.OpenDatabase (stConnect, stOwner, stQualifier)

Part	Description
gc	A GuardCardCtrl variable.
stConnect	[in] A connection string that specifies the ODBC or JET database to open.
stOwner	[in] The database owner.
stQualifier	[in] The database qualifier string.

EnableQueries

Call this function to enable or disable the users ability to execute queries.

Syntax

gc.EnableQueries(fEnable)

Part	Description
gc	A GuardCardCtrl variable.
fEnable	[in] Boolean value: True to enable queries, False to disable queries.

EnableErrorMessages

By default, GuardCard will not display error messages to the user when operations are performed through the automation interface. The error messages will be returned to the client application when an error occurs. If error messages are enabled, the user will be required to respond to any error messages before control will return to the calling application. This behavior can be enabled or disabled by calling this function.

Syntax

gc.EnableErrorMessages(fEnable)

Part	Description
gc	A GuardCardCtrl variable.

fEnable	[in] Boolean value: True to display error messages, False to suppress them.
---------	---

SelectPersonRecord

Display a person record.

Syntax

gc.SelectPersonRecord (stPersonID)

Part	Description
gc	A GuardCardCtrl variable.
stPersonID	[in] Person_ID of the person record to display.

SelectCardRecord

Display a card record.

Syntax

gc.SelectCardRecord (stCardID)

Part	Description
gc	A GuardCardCtrl variable.
stCardID	[in] Card_ID of the card record to display.

EditPersonRecord

Display a person record and make it editable. This function only applies if the user level is “Clerk” or lower. Higher user levels can edit all records by default. Call this function to allow the user to edit one record only.

Syntax

gc.EditPersonRecord (stPersonID)

Part	Description
gc	A GuardCardCtrl variable.
stPersonID	[in] Person_ID of the person record to display.

EditCardRecord

Display the card record specified by stCardID and make it editable. This function only applies if the user level is “Clerk” or lower. Higher user levels can edit all records by default. Call this function to allow the user to edit one record only.

Syntax

gc.EditCardRecord (stCardID)

Part	Description
gc	A GuardCardCtrl variable.
stCardID	[in] Card_ID of the card record to display.

CheckCard

Check if a card is a “Valid” card, where valid means the card status is Valid, and the current date and time falls in the range ‘Start Date’ to ‘End Date’. Start Date and End Date are defined in the Card table. CheckCard will display the card and associated person record of the card being checked, if it can be found.

Syntax

```
ret = gc.CheckCard(stCardCode)
```

Part	Description
ret	[retval] Boolean value: True if the card is ‘Valid’, False otherwise.
gc	A GuardCardCtrl variable.
stCardCode	[in] The card code of the card to check.

ExecuteQuery

Execute a query for person and/or card records.

Syntax

```
gc.ExecuteQuery(stPersonCrit, stCardCrit, stPersonSort, stCardSort)
```

Part	Description
gc	A GuardCardCtrl variable.
stPersonCrit	[in] Person Criteria
stCardCrit	[in] Card Criteria
stPersonSort	[in] Person sort order
stCardSort	[in] Card sort order

Remarks

The four criteria above are used to select the set of records to export. All, none, or any combination of these criteria can be specified. The strings must be syntactically valid SQL strings for each of these criteria.

An example of a person criterion is:

Last_Name='Dupont'.

An example of a person sort order is:

Last_Name DESC.

If stPersonCrit is empty, all person records will be selected; if stCardCrit is empty, all card records will be selected. If both stPersonSort and stCardSort are empty, the default person and card sort order will be used.

EnumNamedQueries

This function returns the list of named queries from GuardCard, such as the ones displayed in the GuardCard Query dropdown list. These names can be used as arguments to the ExecuteNamedQuery method.

Syntax

```
ret = gc.EnumNamedQueries
```

Part	Description
------	-------------

ret	[retval] A variant containing an array of strings filled with query names.
gc	A GuardCardCtrl variable.

ExecuteNamedQuery

Execute one of the GuardCard named queries.

Syntax

gc.ExecuteNamedQuery (stQueryName)

Part	Description
gc	A GuardCardCtrl variable.
stQueryName	[in] A query name previously defined in GuardCard.

PrintCard

Print the specified card record with or without user intervention. The card record will be displayed before it is printed. The PrintCard method will return an error if any errors occur during the print operation. However, if the print dialog is shown, the user will have the opportunity to cancel the print operation, which will not be considered an error. To determine if a card has actually been printed, client applications should listen for the CardPrinted event.

Syntax

gc.PrintCard (stCardID, fShowDialog)

Part	Description
gc	A GuardCardCtrl variable.
stCardID	[in] The Card_ID of the card record to print.
fShowDialog	[in] Boolean value: True to show the print dialog, False to print without user interaction.

CaptureImageType

Initiate image capture by Image Type ID for the specified person record. This will initiate image capture and wait for the user to interact with the GuardCard UI to complete the capture. Control will not return to the client application until the user has completed or cancelled the capture operation. If the user cancels the Capture operation, an error will not be raised. Client applications should listen for the ImageCaptured event to determine if an image has been captured successfully.

Syntax

gc.CaptureImageType (lImageType, stPersonID)

Part	Description
gc	A GuardCardCtrl variable.
lImageType	[in] A long integer that specifies the image type to capture.
stPersonID	[in] Person_ID of the person record to capture the image for.

CaptureImageName

Initiate image capture by image type name for the specified person record. This will initiate image capture and wait for the user to interact with the GuardCard UI to complete the capture. Control will not return to the client application until the user has completed or cancelled the capture operation. If the user cancels the capture operation, an error will not be raised. Client applications should listen for the ImageCaptured event to determine if an image has successfully been captured.

Syntax

gc.CaptureImageName (stImageType, stPersonID)

Part	Description
gc	A GuardCardCtrl variable.
stImageName	[in] Specifies the image type to capture by name. (ex. Photograph, Fingerprint, etc.)
stPersonID	[in] Person_ID of the person record to capture the image for.

BatchExport

Perform a GuardCard batch export operation. This operation will not require user interaction.

Syntax

gc.BatchExport (stPersonCrit, stCardCrit, stPersonSort, stCardSort, fPersons, fCards, fImages, varCardFields, varPersonFields, varLogPersonFields, fMakeDirs, stDir, stCSVFile, stLogFile, cP, cC, cI, cE)

Part	Description
gc	A GuardCardCtrl variable.
stPersonCrit	[in] Person Criteria
stCardCrit	[in] Card Criteria
stPersonSort	[in] Person sort order
stCardSort	[in] Card sort order

Remarks

The four criteria above are used to select the set of records to export. All, none, or any combination of these criteria can be specified. The strings must be syntactically valid SQL strings for each of these criteria.

An example of a person criterion is:

Last_Name='Dupont'.

An example of a person sort order is:

Last_Name DESC.

If stPersonCrit is empty, all person records will be selected; if stCardCrit is empty, all card records will be selected. If both stPersonSort and stCardSort are empty, the default person and card sort order will be used.

Part	Description
fPersons	[in] Boolean value: True to export person records.
fCards	[in] Boolean value: True to export card records.

fImages	[in] A long integer that indicates which image types to export. Set to 0 for no images, 1 for Photograph, 2 for Signature, and 4 for Fingerprint. Combine values to export multiple image types, for example 7 to export all image types.
varCardFields	[in] An array of strings containing a list of card fields to export.
varPersonFields	[in] An array of strings containing a list of person fields to export.
varLogPersonFields	[in] An array of strings containing two person fields to use in the log file.
fMakeDirs	[in] Boolean value: True to create a subdirectory for each image type exported, False to export all images into the directory specified in stDir. Ignored if no image types are exported.
stDir	[in] The full path of the directory used to store files created during the batch export process.
stCSVFile	[in] The name of the CSV file created during the batch export process. The CSV file is created in the directory specified by stDir.
stLogFile	[in] The name of the log file created during the batch export process. The log file is created in the directory specified by stDir.
cP	[out] A long integer, a count of person records exported.
cC	[out] A long integer, a count of card records exported.
cI	[out] A long integer, a count of images exported.
cE	[out] A long integer, a count of records that caused errors while attempting to export.

ActivateGuardCard

Bring the GuardCard window to the front and give it the input focus. Also, if the GuardCard window is minimized, restore it to normal size.

Syntax

gc.ActiveGuardCard()

Part	Description
gc	A GuardCardCtrl variable.

ExitGuardCard

Quit the GuardCard application. All client applications connected to GuardCard are disconnected.

Syntax

gc.ExitGuardCard (fAskUser)

Part	Description
gc	A GuardCardCtrl variable.
fAskUser	[in] Boolean value: True to ask user whether or not to save unsaved records before closing, False to save any changes without first asking the user. If fAskUser is True, the user can cancel the shutdown operation, if there are unsaved records.

Method Table

The following tables details four properties of the IGuardCardCtrl methods:

- Show Window – Methods that require interaction from the user in GuardCard will first bring the GuardCard to the foreground.
- Database Required - Indicates if an open database is required to execute the method successfully. If no database is open the *EPIAutoErrNoDB* error will be raised.
- Synchronous - Most of the methods are synchronous, meaning that control will not return to the client until the operation is completed. Exceptions to this rule are *ExitGuardCard*, *EditCardRecord* and *EditPersonRecord*. *ExitGuardCard* sends a close message to the application window and returns before the application has actually closed. *EditCardRecord* and *EditPersonRecord* put GuardCard in a state that enables the user to edit the record and then returns before the user has edited the record.
- Security Check – some GuardCard functionality can only be executed by users with the appropriate user security level (See IGuardCardCtrl::*EnableSecurityOverride*). If the current user does not have the security level required, the *EPIAutoErrSecurity* error will be raised.

Method Name	Show Window	Database Required	Synchronous	Security Check
GetRunState	No	No	Yes	No
GetCurrentUser	No	No	Yes	No
Login	No ²	No	Yes	No
EnableSecurityOverride	No	No	Yes	No
DisableSecurityOverride	No	No	Yes	No
EnableQueries	No	No	Yes	No
EnableErrorMessage	No	No	Yes	No
OpenDatabase	No ¹	No	Yes	No
SelectPersonRecord	No ²	Yes	Yes	No
SelectCardRecord	No ²	Yes	Yes	No
EditPersonRecord	No ²	Yes	No	Yes ⁵
EditCardRecord	No ²	Yes	No	Yes ⁵
CheckCard	No ²	Yes	Yes	No
ExecuteQuery	No ²	Yes	Yes	No

EnumNamedQueries	No	Yes	Yes	No
ExecuteNamedQuery	No ²	Yes	Yes	No
PrintCard	Yes/No ^{2,3}	Yes	Yes	Yes
CaptureImageType	Yes	Yes	Yes	Yes
CaptureImageName	Yes	Yes	Yes	Yes
BatchExport	No ²	Yes	Yes	Yes
ActivateGuardCard	Yes	No	Yes	No
ExitGuardCard	Yes/No ⁴	No	No	No

¹ *OpenDatabase* may require user interaction, for example to enter a user name and password for the database. If this is the case, *ActivateGuardCard* should first be called to bring the GuardCard window to the foreground.

² Methods that require GuardCard to change the displayed card and/or person record(s) will prompt the user if there are any unsaved records. Before calling any of these methods *ActivateGuardCard* should be called unless the caller is sure there are no records with unsaved changes displayed.

³ The GuardCard window will only be brought to the foreground when *PrintCard* is called with *fShowDialog* set to True.

⁴ The GuardCard window will only be brought to the foreground when *ExitGuardCard* is called with *fAskUser* set to True.

⁵ Requires a security override.

IGuardCardEvents

The purpose of the event interface is to provide a small set of useful events, one or more of which could be used by a third party application to integrate GuardCard functionality into their system.

Events are triggered when the following actions are performed in GuardCard:

- Display, Add, Update, or Delete a Person Record
- Display, Add, Update, or Delete a Card Record
- Add, Update, or Delete an Image
- Change card status
- Print card

Note that the purpose of this event interface is not to provide an exhaustive set of events that would allow the client application to trace every action taken by the GuardCard operator. For example, although a *CardDeleted* event is provided, there are several ways to delete cards in GuardCard, and only the deletion of individual card records will trigger an event. Multiple cards can be deleted through actions such as Batch Delete. Such actions would not trigger the *CardDeleted* event(s). In general, events are triggered in response to user actions on single records. The exception to this is *BatchPrint*. A *CardPrinted* event is triggered for each card during a Batch Print operation.

Events

BeforePersonAdded

This event is sent just before a new person record is saved in the database.

Syntax

```
private sub gc_ BeforePersonAdded(stUser, vData, fContinue)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
vData	[in] A GAFieldSet object containing the record's data.
fContinue	[out] A boolean value. If the application sets it to False, the operation will be aborted and the record will not be saved in the database.

BeforePersonUpdated

This event is sent just before an existing person record is updated.

Syntax

```
private sub gc_ BeforePersonUpdated(stUser, stPersonID, vData, fContinue)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record.
vData	[in] A GAFieldSet object containing the record's data.
fContinue	[out] A boolean value. If the application sets it to False, the operation will be aborted and the record will not be saved in the database.

BeforeCardAdded

This event is sent just before a new card record is saved in the database.

Syntax

```
private sub gc_ BeforeCardAdded(stUser, vData, fContinue)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
vData	[in] A GAFieldSet object containing the record's data.
fContinue	[out] A boolean value. If the application sets it to False, the operation will be aborted and the record will not be saved in the database.

BeforeCardUpdated

This event is sent just before an existing card record is updated.

Syntax

```
private sub gc_ BeforeCardUpdated(stUser, stCardID, vData, fContinue)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
vData	[in] A GAFieldSet object containing the record's data.
fContinue	[out] A boolean value. If the application sets it to False, the operation will be aborted and the record will not be saved in the database.

BeforeImageAdded

This event is sent just before a new image is saved in the database.

Syntax

```
private sub gc_ BeforeImageAdded(stUser, stPersonID, lImageType, vImage, fContinue)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record that contains the image.
lImageType	[in] A long integer that specifies the image type of the added image.
vImage	[in] A GAImage object containing the image.
fContinue	[out] A boolean value. If the application sets it to False, the operation will be aborted and the record will not be saved in the database.

BeforeImageUpdated

This event is sent just before an image is saved in the database to replace an existing one.

Syntax

```
private sub gc_ BeforeImageUpdated(stUser, stPersonID, lImageType, vImage, fContinue)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record that contains the image.
lImageType	[in] A long integer that specifies the image type of the updated image.
vImage	[in] A GAImage object containing the image.

fContinue	[out] A boolean value. If the application sets it to False, the operation will be aborted and the record will not be saved in the database.
-----------	---

PersonDisplayed

This event is triggered when a person record is displayed through any user interaction, such as scrolling through records, executing a query, and so forth.

Syntax

```
private sub gc_PersonDisplayed(stUser, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record.

PersonAdded

This event is triggered when a new person record is saved.

Syntax

```
private sub gc_PersonAdded(stUser, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record.

PersonUpdated

This event is triggered when changes are saved to a person record.

Syntax

```
private sub gc_PersonUpdated(stUser, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record.

PersonDeleted

This event is triggered when a person record is deleted.

Syntax

```
private sub gc_PersonDeleted(stUser, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record.

ImageAdded

This event is triggered when a new image is captured.

Syntax

```
private sub gc_ImageAdded(stUser, stPersonID, lImageType, stFileName)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record that contains the image.
lImageType	[in] A long integer that specifies the image type of the added image.
stFileName	[in] The full path and filename of the image that was added. This parameter is valid only when the “Store Images in Files” option is set in the GuardCard image setup dialog.

ImageUpdated

This event is triggered when an image is updated, either by modifying the existing image or capturing a new image to replace the existing one.

Syntax

```
private sub gc_ImageUpdated(stUser, stPersonID, lImageType, stFileName)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record that contains the image.
lImageType	[in] A long integer that specifies the image type of the updated image.
stFileName	[in] The full path and filename of the image that was modified. This parameter is valid only when the “Store Images in Files” option is set in the GuardCard image setup dialog.

ImageDeleted

This event is triggered when an image is deleted.

Syntax

```
private sub gc_ImageDeleted(stUser, stPersonID, lImageType)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stPersonID	[in] The value of the Person_ID field for the person record that contains the image.

lImageType	[in] A long integer that specifies the image type of the deleted image.
------------	---

CardDisplayed

This event is triggered when a card record is displayed through any user interaction, such as scrolling through records, executing a query, and so forth.

Syntax

```
private sub gc_CardDisplayed(stUser, stCardID, stCardCode, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
stCardCode	[in] The value of the Card_Code field for the card record.
stPersonID	[in] The value of the Card_Person_ID field of the card record.

CardAdded

This event is triggered when a new card record is saved.

Syntax

```
private sub gc_CardAdded(stUser, stCardID, stCardCode, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
stCardCode	[in] The value of the Card_Code field for the card record.
stPersonID	[in] The value of the Card_Person_ID field of the card record.

CardUpdated

This event is triggered when changes are saved to a card record.

Syntax

```
private sub gc_CardUpdated(stUser, stCardID, stCardCode, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
stCardCode	[in] The value of the Card_Code field for the card record.
stPersonID	[in] The value of the Card_Person_ID field of the card record.

CardDeleted

This event is triggered when a card record is deleted.

Syntax

```
private sub gc_CardDeleted(stUser, stCardID, stCardCode, stPersonID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
stCardCode	[in] The value of the Card_Code field for the card record.
stPersonID	[in] The value of the Card_Person_ID field of the card record.

CardStatusChanged

There are three ways a card status can be changed. Two ways are manual: 1) Validate Printed Cards, 2) Change Card Status. It is also automatically updated during the print process. This event will be triggered only for manual card status changes, not as the result of a print operation.

Syntax

```
private sub gc_CardStatusChanged(stUser, stCardID, stCardCode, stPersonID, lOldStatus, lNewStatus)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
stCardCode	[in] The value of the Card_Code field for the card record.
stPersonID	[in] The value of the Card_Person_ID field of the card record.
lOldStatus	[in] A long integer that specifies what the card status was before it was changed.
lNewStatus	[in] A long integer that specifies what the new card status is.

CardPrinted

This event is triggered after a card is successfully printed.

Syntax

```
private sub gc_CardPrinted(stUser, stCardID, stCardCode, stPersonID, lOldStatus, lNewStatus, stFormatID)
```

Part	Description
stUser	[in] The User ID of the current GuardCard user.
stCardID	[in] The value of the Card_ID field for the card record.
stCardCode	[in] The value of the Card_Code field for the card record.

stPersonID	[in] The value of the Card_Person_ID field of the card record.
lOldStatus	[in] A long integer that specifies what the card status was before it was printed.
lNewStatus	[in] A long integer that specifies the new card status after printing completed.
stFormatID	[in] The Card_Format_ID of the printed card.

GuardCardExited

This event is triggered just before GuardCard finishes shutting down. At this point the user cannot cancel the shutdown of GuardCard.

Syntax

```
private sub gc_GuardCardExited()
```

GTImporterCtrl Object

The GTImporterCtrl object is used to control and receive events from the GuardTool Importer utility. GTImporterCtrl implements the IGTImporterCtrl interface and acts as a source of events for the IGTImporterEvents interface.

IGTImporterCtrl

Provides methods to start the GuardTool Importer, and import a set of records using a previously defined importer definition file. During the import process, events are triggered as records are added, updated or rejected.

Methods

GetRunState

Call this function to determine how GuardTool Importer was started; either programmatically, or by the user. When a GTImporterCtrl automation object is created, it will cause GT Importer to startup if an instance was not already running. Note: There is no Start method in the IGTImporterCtrl interface - creating a GTImporterCtrl object automatically starts the application. This function is provided to allow the client application the option to take control of a user initiated GT Importer session.

Syntax

```
ret = gti.GetRunState
```

Part	Description
ret	[retval] An EPIRunState constant. A value of EPIRunAutomation indicates GT Importer was started programmatically, a value of EPIRunUser indicates GT Importer was started by the user.
gti	A GTImporterCtrl variable.

Login

Login to GuardTool Importer. Only Administrator-level users can log in to GuardTool Importer.

Syntax

```
gti.Login(stUser, stPassword)
```

Part	Description
gti	A GTImporterCtrl variable.
stUser	[in] User ID.
stPassword	[in] Password.

Import

Import records based on a previously defined importer .def file. During the importing process the Importer will trigger events as records are added, updated or rejected. These events are defined in the IGTImporterEvents interface below.

Syntax

```
gti.Import(stDefFileName, stLogFileName, fAppendLog)
```

Part	Description
gti	A GTImporterCtrl variable.
stDefFileName	[in] The full path and filename of the .def file to use.
stLogFileName	[in] The full path and filename of the log file created during the import process.
fAppendLog	[in] Boolean value: True to append to the log file if it already exists, False to overwrite it.

ExitGTImporter

This method shuts down the GT Importer application. Any connected clients will be disconnected.

Syntax

```
gti.ExitGTImporter()
```

Part	Description
gti	A GTImporterCtrl variable.

IGTImporterEvents

An event is triggered for each person, card, or image record during the import process.

PersonAdded

This event is triggered after a new person record has been added to the destination database.

Syntax

```
private sub gti_PersonAdded(stPersonID, varPMKFields, varPMKValues)
```

Part	Description
stPersonID	[in] Person_ID for the newly created person record.
varPMKFields	[in] An array of strings containing the field names of the person primary key from the source database.
varPMKValues	[in] An array of strings containing the values of the person primary key fields from the source database. Ordered the same as the field names.

PersonUpdated

This event is triggered after a person record has been updated in the destination database.

Syntax

```
private sub gti_PersonUpdated(stPersonID, varPMKFields, varPMKValues)
```

Part	Description
stPersonID	[in] Person_ID of the updated person record.

varPMKFields	[in] An array of strings containing the field names of the person primary key from the source database.
varPMKValues	[in] An array of strings containing the values of the person primary key fields from the source database. Ordered the same as the field names.

PersonError

This event is triggered after an error has occurred while importing a person record.

Syntax

```
private sub gti_PersonError(nError, varPMKFields, varPMKValues)
```

Part	Description
nError	[in] A long integer containing an error code, always 0.
varPMKFields	[in] An array of strings containing the field names of the person primary key from the source database.
varPMKValues	[in] An array of strings containing the values of the person primary key fields from the source database. Ordered the same as the field names.

CardAdded

This event is triggered after a new card record has been added to the destination database.

Syntax

```
private sub gti_CardAdded(stCardID, stCardCode, stPersonID, varPMKFields, varPMKValues,  
varFKFields, varFKValues)
```

Part	Description
stCardID	[in] Card_ID of the newly created card record in the destination database.
stCardCode	[in] Card_Code of the newly created card record in the destination database.
stPersonID	[in] Card_Person_ID for the newly created card record.
varPMKFields	[in] An array of strings containing the field names of the card primary key from the source database.
varPMKValues	[in] An array of strings containing the values of the card primary key fields from the source database. Ordered the same as the field names.
varFKFields	[in] An array of strings containing the field names of the card foreign key of the source database.
varFKValues	[in] An array of strings containing the values of the card foreign key fields of the source database. Ordered the same as the field names.

CardUpdated

This event is triggered after a card record has been updated in the destination database.

Syntax

```
private sub gti_CardUpdated(stCardID, stCardCode, stPersonID, varPMKFields, varPMKValues,
varFKFields, varFKValues)
```

Part	Description
stCardID	[in] Card_ID of the updated card record in the destination database.
stCardCode	[in] Card_Code of the updated card record in the destination database.
stPersonID	[in] Card_Person_ID of the updated card record in the destination database.
varPMKFields	[in] An array of strings containing the field names of the card primary key from the source database.
varPMKValues	[in] An array of strings containing the values of the card primary key fields from the source database. Ordered the same as the field names.
varFKFields	[in] An array of strings containing the fields names of the card foreign key of the source database.
varFKValues	[in] An array of strings containing the values of the card foreign key fields of the source database. Ordered the same as the field names.

CardError

This event is triggered after an error has occurred while importing a card record.

```
private sub gti_CardError(nError, varPMKFields, varPMKValues, varFKFields, varFKValues)
```

Part	Description
nError	[in] A long integer containing an error code, always 0.
varPMKFields	[in] An array of strings containing the field names of the card primary key from the source database.
varPMKValues	[in] An array of strings containing the values of the card primary key fields from the source database. Ordered the same as the field names.
varFKFields	[in] An array of strings containing the fields names of the card foreign key of the source database.
varFKValues	[in] An array of strings containing the values of the card foreign key fields of the source database. Ordered the same as the field names.

ImageAdded

This event is triggered after a new image has been added to the destination database.

Syntax

```
private sub gti_ImageAdded(lImageType, stPersonID, varPMKFields, varPMKValues)
```

Part	Description
lImageType	[in] A long integer that specifies the image type of the newly added image.
stPersonID	[in] The Person_ID of the person record containing the new image in the destination database.
varPMKFields	[in] An array of strings containing the field names of the source database person primary key associated with the image.
varPMKValues	[in] An array of strings containing the values of the source database person primary key fields associated with the image. Ordered the same as the field names.

ImageUpdated

This event is triggered after an image has been updated in the destination database.

Syntax

```
private sub gti_ImageUpdated(lImageType, stPersonID, varPMKFields, varPMKValues)
```

Part	Description
lImageType	[in] A long integer that specifies the image type of the updated image.
stPersonID	[in] The Person_ID of the person record containing the updated image in the destination database.
varPMKFields	[in] An array of strings containing the field names of the source database person primary key associated with the image.
varPMKValues	[in] An array of strings containing the values of the source database person primary key fields associated with the image. Ordered the same as the field names.

ImageError

This event is triggered after an error has occurred while importing an image.

Syntax

```
private sub gti_ImageError(nError, varPMKFields, varPMKValues, stImageField)
```

Part	Description
nError	[in] A long integer containing an error code, always 0.
varPMKFields	[in] An array of strings containing the field names of the source database person primary key associated with the image.

varPMKValues	[in] An array of strings containing the values of the source database person primary key fields associated with the image. Ordered the same as the field names.
stImageField	[in] The name of the image field from the source database.

GTImporterExited

This event is triggered just before GTImporter finishes shutting down.

Syntax

```
private sub gti_GTImporterExited()
```

Error Handling

When a method fails, an error is raised containing an error code and error string. The client application is responsible for handling errors by presenting error dialogs to the user, logging the error in error log files, or by taking exceptional action. For example, calling “On Error GoTo line” statement in Visual Basic causes the code to jump to the specified line when errors (returned from EPI Suite) are trapped. Visual Basic provides both the number and description of the error through the Err and Error functions.

The following is a list of error codes defined for EPI Suite OLE Automation components. In addition to these, other COM error codes (such as E_INVALID_ARG) may be returned.

- EPIAutoErrFailed - operation failed
- EPIAutoErrSecurity - current user security level insufficient
- EPIAutoErrBadRecord - an invalid or missing record was specified
- EPIAutoErrInternal - an internal application error occurred
- EPIAutoErrUnsavedRecs - there are unsaved records
- EPIAutoErrNoRecords - a query returned no records
- EPIAutoErrBusy - application is busy performing another task
- EPIAutoErrNoLogin - no user logged in to application
- EPIAutoErrInvalidCard - invalid card record specified
- EPIAutoErrInvalidPerson - invalid person record specified
- EPIAutoErrInvalidType - invalid image type specified
- EPIAutoErrNoDB - no open database
- EPIAutoErrInvalidUser - invalid user name or password specified
- EPIAutoErrNotAdmin - operation requires an Administrator-level user
- EPIAutoErrBadFileOrDir - a file error occurred
- EPIAutoErrBadQueryName - the specified query name is invalid