INDEX

1. QckPlugin Framework Overview
2. Glossary of Key Terms
    a. QckPlugin
    b. QckService
    c. QckExtension
    d. QckExtendedService
    e. QckEvent
    f. QckEventHandler
    g. QckEvent Notification
3. API Reference
    a. Functions
        1. InitQckPlugin
        2. ExecuteQckPlugin
        3. CloseQckPlugin
    b. Classes
        1. IQckSrvcMgr
            a. RegisterForWindowService
            b. RegisterForMenuItemService
            c. RegisterForMarkerService
            d. RegisterForExtractService
            e. RegisterForGraphicsService
            f. RegisterForQueryService
            g. RegisterForDatabaseService
            h. RegisterForLayerService
            i. AddExtendedService
            j. GetExtendedService
            k. GetNotificationHandler
            l. GetLastErrorCode
            m. GetLastErrorMsg
        2. IQckWindowSrvc
            a. SetWindowServiceNotify
            b. SelectPoint
            c. SetDrawRubberBand
            d. SetWindow
            e. GetCurrentWindow
            f. SelectPolygonArea
        3. IQckMenuSrvc
            a. CreateMenuItem
            b. CreatePopupMenu
            c. AddPopupMenuItem
            d. GetAppShell (UNIX/Linux only)
        4. IQckMarkerSrvc
            a. DrawMarker
            b. ZoomToMarker
        5. IQckExtractSrvc
            a. ExtractGDSII (Rectangular Window)
            b. ExtractGDSII (Polygonal Window)
        6. IQckGraphicsSrvc
            a. SetDrawServiceNotify
            b. Redraw
            c. SelectForegroundColor
            d. SelectColor

e. DrawBoundary
f. IsSmallObject
g. SetLineSize
h. GetPixelSize
7. IQckQuerySrvc
a. Stop
b. SetQueryServiceNotify
c. SetViewerInfoMode
d. SetReferenceVectorMatrix
e. GetDataVector
f. GetCellReferences
8. IQckDatabaseSrvc
a. SetDatabaseServiceNotify
b. SetViewCell
c. GetGrid
d. GetUnits
e. GetCellID
f. GetCellName
g. GetViewCell
h. GetCellList
i. GetCellRoot
j. GetCellParents
k. GetCellChildren
l. GetCellExtents
m. GetCellVertices
n. GetCellInfo
o. GetLayerList (as shorts)
p. GetLayerList (as strings)
q. GetDatatypesForLayer
r. FreeCellList
s. FreeLayerList (shorts)
t. FreeLayerList (strings)
9. IQckLayerSrvc
a. SetLayerServiceNotify
b. SetLayerOnOff
c. SetAllLayersOnOff
d. GetLayerOnOff
10. IQckEventNotify
a. OnDrawWindowRubberBand
b. OnChangeViewWindow
c. OnSelectPoint
d. OnRedraw
e. OnGetDataVector
f. OnInfoObject
g. PreFileOpen
h. PostFileOpen
i. PreCellOpen
j. PostCellOpen
k. OnSelectPolygonArea
l. OnChangeLayers
m. OnShowDialog (WINDOWS only)
c. Data Structures
1. sQCKPLUGINARGS

2. sQCKWINDOW
3. sQCKPOLYWINDOW
4. sQCKMARKER
5. sQCKEXTRACTPARAMS
6. sQCKEXTRACTWINDOW
7. sQCKEXTRACTPOLY
8. sQCKBOUNDARY
9. sQCKPATH
10. sQCKSREF
11. sPARENTREF
12. sQCKHSREF
13. sTMATRIX
14. sTPARAMS
15. sQCKAREF
16. sQCKHAREF
17. sQCKTEXT
18. sQCKVECTOR
19. sQCKINFOVECTOR
20. sQCKQUERYSETTINGS

d. Flags

e. ErrorCodes

4. Version History
   - v1.1.0 (Mar 28, 2009)
   - v1.2.0 (Apr 02, 2009)
   - v1.3.0 (Apr 10, 2009)
   - v1.4.0 (May 21,2009)
   - v1.5.0 (May 28, 2009)

QCKPLUGIN FRAMEWORK OVERVIEW

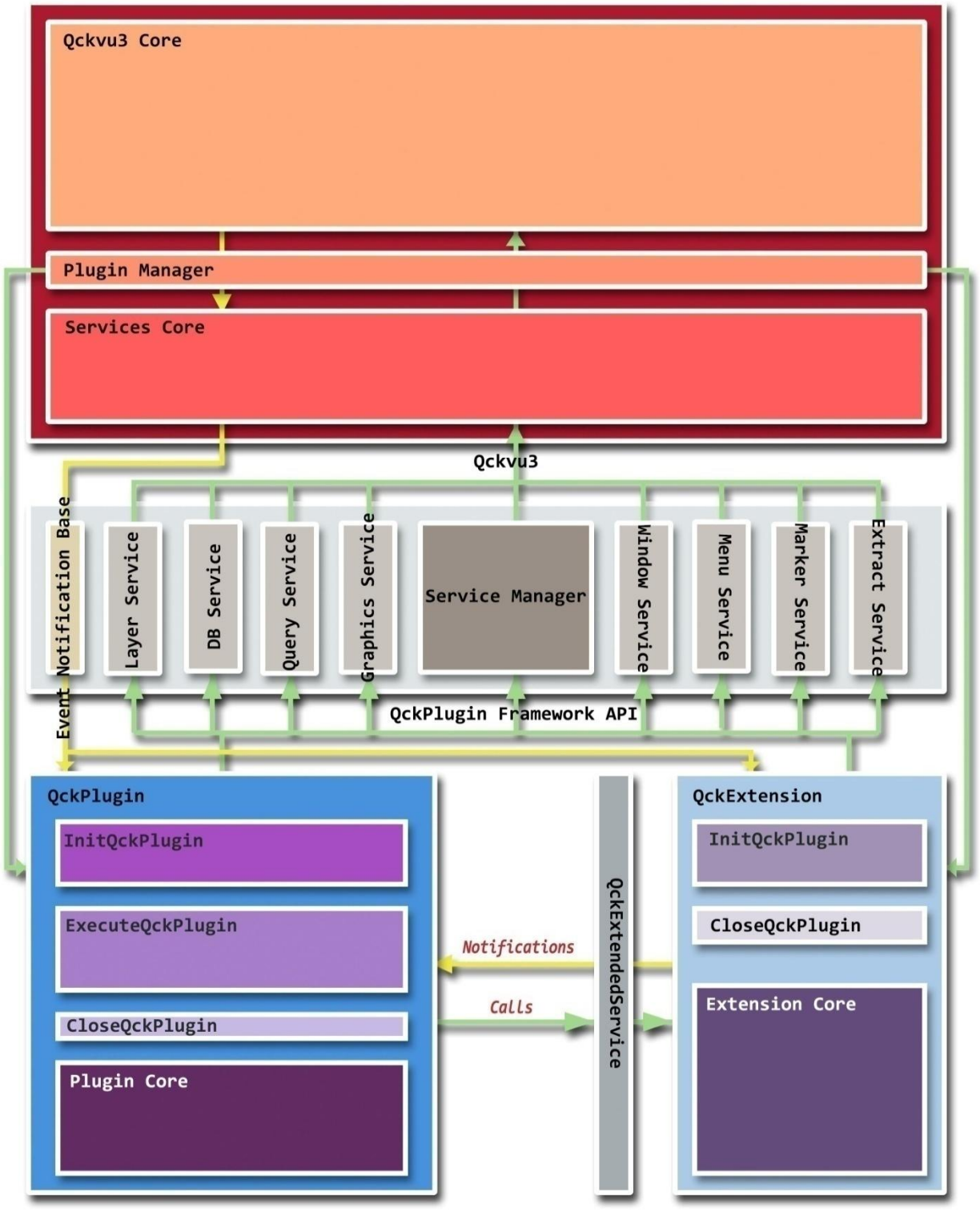

Fig 1

GLOSSARY OF KEY TERMS

QckPlugin

- A QckPlugin is a plug-in (a shared library, usually a .dll on Windows and .so on Unix/Linux) that conforms to the QckPlugin Framework API.
- A QckPlugin can be loaded into Qckvu3 dynamically at run-time and it's presence or absence does not affect the primary behavior of Qckvu3 (to work as a CAD database viewer).
- A plug-in must satisfy the following conditions to qualify as a QckPlugin :-
  1. It must be a valid shared library (with extensions .dll on Windows, .so on Unix/Linux).
  2. It must define the InitQckPlugin and ExecuteQckPlugin functions.
  3. It must be placed at a suitable location relative to the Qckvu3 executable so that Qckvu3 can detect and load it at run-time.
- At present, Qckvu3 supports only one mechanism to detect plug-ins. All plug-ins must be placed in a location '<execDir>/plugins/<pluginName>' where execDir is the directory containing the Qckvu3 binary, pluginName is a directory that contains the plug-in and files needed by the plug-in.
- At present, Qckvu3 loads plug-ins only at start-up.
- A QckPlugin is a binary (file).

QckService

- A QckService is a set of API functions (interface) made available to a QckPlugin so that it can communicate with Qckvu3 and utilize the various functionalities that Qckvu3 provides to plug-ins.
- Each QckService is represented by an interface (abstract) class containing the relevant methods. e.g The QckWindow service is represented by the class IQckWindowSrvc.
- This interface class is declared in the header file 'qckplugin.h' which is made available to the plug-in writer as a part of the Qckvu3 package.
- To access a service, a plug-in requires a handle to an object that implements the service interface. This handle can be obtained by calling the approprite 'Register' functions of the IQckSrvcMgr (Service Manager) interface class. e.g to get access to the QckWindow service, the plug-in must call RegisterForWindowService method of the IQckSrvcMgr class.
- The handle to the Service Manager is made available to a plug-in through the InitQckPlugin function when the plug-in is being loaded at start-up.
- A QckService is an interface (class).

QckExtension

- A QckExtension is a special plug-in that provides back-end service(s) to other QckPlugins in a way extending the existing QckPlugin Framework API.
- Just like regular QckPlugins, a QckExtension is also loaded at start-up.
- A plug-in must satisfy the following conditions to qualify as a QckExtension :-
  1. It must be a valid shared library (with extensions .dll on Windows, .so on Unix/Linux).
  2. It must define the InitQckPlugin function.
  3. It must register its services my calling the AddExtendedService method of the IQckSrvcMgr class during the InitQckPlugin function call.
  4. It must be placed at a suitable location relative to the Qckvu3 executable so that Qckvu3 can detect and load it at run-time.
- At present, Qckvu3 supports only one mechanism to detect plug-ins. All plug-ins must be placed in a location '<execDir>/plugins/<pluginName>' where execDir is the directory containing the Qckvu3 binary, pluginName is a directory that contains the plug-in and files needed by the plug-in.
- Because a QckExtension is loaded at start-up indifferent from any other QckPlugin, it's services can

be accessed only during the ExecuteQckPlugin function.
- Each QckExtension must be accompanied by a header file that declares the various interface classes for the QckExtendedServices it offers.
- A QckExtension is a binary (file).

QckExtendedService

- A QckExtendedService is a set of API functions provided by a QckExtension.
- Just like a QckService, it consists of a set of related functions and is represented by an interface class.
- A QckExtendedService interface class is present in a header file accompanying the QckExtension in which that service is implemented.
- Each QckExtendedService must have a unique service name that identifies it from other QckExtendedServices. This name must also be made available in the same header file that contains the interface class.
- A QckExtendedService can be made available to other QckPlugins only if it is registered with the QckPlugin Framework. To do so, the QckExtension must call the AddExtendedService method of the IQckSrvcMgr (Service Manager) class during the InitQckPlugin function call.
- A QckPlugin can access a QckExtendedService by means of a handle to an object that implements the service interface. This object is present inside the corresponding QckExtension binary.
- A QckPlugin can get access to such a handle by calling the GetExtendedService method of the IQckSrvcMgr (Service Manager) interface class. It must use the QckExtendedService name to request access to that service.
- A QckExtendedService is an interface (class).

QckEvent

- A Qckvu3 Event occurs when a user performs an action on Qckvu3 using one of the many controls available through the Qckvu3 graphical user interface.
- A QckEvent is a subset of Qckvu3 events that provide notifications (when the event occurs) to candidate QckPlugins via the QckPlugin Framework API.
- The QckPlugin Framework provides notifications by invoking QckEventHandlers defined inside the plug-in.
- A QckPlugin can choose which notifications to receive both at compile-time and run-time.

QckEventHandler

- A QckEventHandler is a run-time object present inside the QckPlugin.
- This object belongs to a class derived from IQckEventNotify.
- The purpose of a QckEventHandler is to receive (handle) notifications corresponding to QckEvents of interest.
- A QckEventHandler can choose at compile-time and run-time which notifications it will receive.
- In order to receive a notification for a QckEvent, the QckEventHandler must over-ride (redefine) the method of IQckEventNotify class that corresponds to that notification. Notifications corresponding to methods of the IQckEventNotify class that have not been over-ridden (redfined) will not be received by the QckEventHandler. This is the compile-time control.
- At run-time, the QckPlugin can call one of the many 'Set<Service>Notify' methods corresponding to the appropriate 'Service' and specify a QckEventHandler to receive notifications from that 'Service'.
- e.g Use SetWindowServiceNotify to receive notifications for QckWindow service events such as

OnChangeViewWindow.

- If no such QckEventHandler is specified (null parameter is passed), then the QckPlugin will not receive notifications corrsponding to that 'Service'. This is the run-time control.
- One QckPlugin can have more than one QckEventHandlers to handle notifications from different services.
- In this way, a QckPlugin can have complete control on which notifications it should receive.
- If there are more than one QckPlugins active, each receives notifications depending on their interest.

QckEvent Notification

- A QckEvent Notification is the mechanism by which Qckvu3 informs QckPlugins of certain events as they happen.
- Refer to QckEvent and QckEventHandler to understand the QckPlugin notification system.

Function	**InitQckPlugin**
Objective:	Invoked by Qckvu3 when it loads a plug-in during start-up. This function must be defined inside a plug-in for it to be qualified as a QckPlugin or a QckExtension.
Prototype:	`int InitQckPlugin(const char* plgPath, void* plgHandle, void* srvcMgr);`
Parameters:	plgPath — Complete path of this QckPlugin. plgHandle — A unique handle to this QckPlugin used to identify this QckPlugin from others. srvcMgr — Handle to the QckService manager. (type IQckSrvcMgr*)
Return Values:	0 — Inform Qckvu3 to load this QckPlugin (success). non-zero value — Inform Qckvu3 to NOT load this QckPlugin because of internal failure during initialization.
Notes:	1. This function must be defined inside all QckPlugins and QckExtensions. 2. At start-up (before Qckvu3 GUI is visible to the user), Qckvu3 loads the plug-ins (QckPlugins and QckExtensions) by calling this function defined inside each of them. 3. If this function is not found inside a plug-in, it is not loaded. 4. This function can be used for the following purposes:- a) To initialize the plug-in components and data structures. b) For QckPlugins, this function must be used to add a menu item to the Qckvu3 menu bar (using the QckMenu Service) from which the user will be able to launch the plug-in. c) For QckExtensions, this function must be used to register the QckExtended Service provided by the plug-in, with the Qckvu3 service manager (using the AddExtendedService) method of the QckService Manager. 5. The function parameters plgPath, plgHandle and srvcMgr must be remembered inside the plug-in so that it can be used later. 6. If this function returns 0, Qckvu3 will load the corresponding plug-in. If the return is non-zero, Qckvu3 will unload the corresponding plug-in and treat it as plug-in initialization error. 7. The order in which the plug-ins are loaded cannot be determined before all plug-ins have been loaded. Therefore, any functionality that depends on the specific order in which the plug-ins are loaded must not be executed during the InitQckPlugin function. e.g The GetExtendedService method must not be used during the InitQckPlugin function because the corresponding QckExtension might not be loaded yet.
Last Revision:	0x5622522
Sample Code:	

Function	**ExecuteQckPlugin**
Objective:	Invoked by Qckvu3 when the user clicks on a menu item corresponding to a QckPlugin. This function must be defined inside a plug-in to be qualified as a QckPlugin.
Prototype:	`void ExecuteQckPlugin(int argc, void* argv);`
Parameters:	argc Number of arguments contained by argv. argv An array of objects of type sQCKPLUGINARGS which contains arguments for the QckPlugin.
Return Values:	none.
Notes:	1. This function must be defined inside all QckPlugins but may not be defined inside a QckExtension. 2. This function is invoked by Qckvu3 when a user clicks the menu item (added by the corresponding QckPlugin during the InitQckPlugin function) associated with the corresponding QckPlugin. 3. Currently, Qckvu3 allows a user to invoke a QckPlugin only via the Qckvu3 menu bar. 4. Since this function will be called as many times as the user clicks on the menu item, it should be used to execute the core functionality for which the QckPlugin was designed. Usually this involves launching a graphical user interface such as a dialog. Any one time initialization should be dealt with inside the InitQckPlugin function. 5. Since it is possible for one QckPlugin to have multiple features, or dialogs, it may be desirable to add more than one menu item, or perhaps a whole new menu containing multiple menu items to the Qckvu3 menu bar associated with that QckPlugin. In that case, the argc and argv contain useful information about the specific menu-item that caused this function to be invoked, based on which, the QckPlugin can determine the action that needs to be taken. 6. If any of the multiple elements of the argv array is of type argtypeMENUID (argv[i].mType == argTypeMENUID), then the corresponding mArg member would contain the ID of the menu-item that was clicked upon by the user. 7. Since this function is invoked by the Qckvu3 GUI, all plug-ins have been loaded by now, therefore functionality that depends on other plug-ins such as GetExtendedService may be executed during this function call.
Last Revision:	0x5622522
Sample Code:	

Function	**CloseQckPlugin**
Objective:	Invoked by Qckvu3 when it is about to un-load a QckPlugin during exit.
Prototype:	`void CloseQckPlugin();`
Parameters:	none.
Return Values:	none.
Notes:	1. The CloseQckPlugin function may be defined inside all plug-ins (QckPlugin and QckExtension). 2. The CloseQckPlugin is invoked for each plug-in that defines it at exit-time when Qckvu3 is about to close it's components and de-allocate it's resources. 3. On completion of this function, Qckvu3 unloads the plug-in from its memory space. 4. The CloseQckPlugin should be used by the plug-in for purposes of de-allocating resources and gracefully closing the underlying components. 5. Since the order in which the plug-ins will be unloaded cannot be determined, any code that depends on other plug-ins in the system must not be used during this function call. 6. Any code involving QckServices or QckExtendedServices must be avoided as well.
Last Revision:	0x5622522

Sample Code:

```
//---- pluginmain.cpp ----//

#include "qckplugin.h"
#include "cqckplugin.h"

static CQckPlugin thePlugin;

void CloseQckPlugin()
{
   thePlugin.OnCloseQckPlugin();
}

//---- cqckplugin.cpp ----//

#include "cqckplugin.h"

void CQckPlugin::OnCloseQckPlugin()
{
  //free allocated global memory
  //release resource handles
  //release licenses
}
```

Class	**IQckSrvcMgr**	
Objective:	An interface that allows a QckPlugin to access various QckServices.	
Members:	none.	
Methods:	RegisterForWindowService	Request access to the QckWindow Service.
	RegisterForMenuItemService	Request access to the QckMenu Service.
	RegisterForMarkerService	Request access to the QckMarker Service.
	RegisterForExtractService	Request access to the QckExtract Service.
	RegisterForGraphicsService	Request access to the QckGraphics Service.
	RegisterForQueryService	Request access to the QckQuery Service.
	RegisterForDatabaseService	Request access to the QckDatabase Service.
	RegisterForLayerService	Request access to the QckLayer Service.
	AddExtendedService	Register an extension service with the QckPlugin Framework.
	GetExtendedService	Request access to a registered extension service.
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckSrvcMgr :: RegisterForWindowService**
Objective:	Request access to the QckWindow Service.
Prototype:	`IQckWindowSrvc* RegisterForWindowService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckWindow Service (success) null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForWindowService allows a plug-in (QckPlugin or QckExtension) to access the QckWindow Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: RegisterForMenuItemService**	
Objective:	Request access to the QckMenu Service.	
Prototype:	`IQckMenuSrvc* RegisterForMenuItemService(int apiVer = QCKPLUGIN_APIVER);`	
Parameters:	apiVer	Version number of the QckPluginAPI in use.
Return Values:	a valid address null	Handle to the QckMenu Service (success). API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForMenuItemService allows a plug-in (QckPlugin or QckExtension) to access the QckMenu Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions, but it is typically used only during the InitQckPlugin function call.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckSrvcMgr :: RegisterForMarkerService**
Objective:	Request access to the QckMarker Service.
Prototype:	`IQckMarkerSrvc* RegisterForMarkerService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckMarker Service (success). null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForMarkerService allows a plug-in (QckPlugin or QckExtension) to access the QckMarker Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: RegisterForExtractService**
Objective:	Request access to the QckExtract Service.
Prototype:	`IQckExtractSrvc* RegisterForExtractService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckExtract Service (success). null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForExtractService allows a plug-in (QckPlugin or QckExtension) to access the QckExtract Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: RegisterForGraphicsService**
Objective:	Request access to the QckGraphics Service.
Prototype:	`IQckGraphicsSrvc* RegisterForGraphicsService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckGraphics Service (success). null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForGraphicsService allows a plug-in (QckPlugin or QckExtension) to access the QckGraphics Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: RegisterForQueryService**
Objective:	Request access to the QckQuery Service.
Prototype:	`IQckQuerySrvc* RegisterForQueryService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckQuery Service (success). null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForQueryService allows a plug-in (QckPlugin or QckExtension) to access the QckQuery Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: RegisterForDatabaseService**
Objective:	Request access to the QckDatabase Service.
Prototype:	`IQckDatabaseSrvc* RegisterForDatabaseService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckDatabase Service (success). null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForQueryService allows a plug-in (QckPlugin or QckExtension) to access the QckDatabase Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: RegisterForLayerService**
Objective:	Request access to the QckLayer Service.
Prototype:	`IQckLayerSrvc* RegisterForLayerService(int apiVer = QCKPLUGIN_APIVER);`
Parameters:	apiVer — Version number of the QckPluginAPI in use.
Return Values:	a valid address — Handle to the QckLayer Service (success). null — API version number mismatch between Qckvu3 and the QckPlugin.
Notes:	1. The RegisterForLayerService allows a plug-in (QckPlugin or QckExtension) to access the QckLayer Service. 2. If the version of the QckPlugin API with which Qckvu3 was built is different from that of the plug-in, this function will return NULL. 3. If this function fails (returns NULL), call GetLastErrorCode and GetLastErrorMsg to get information about the error. 4. Multiple calls to this function will yeild the same handle (address). 5. The handle returned by this function should be used only during the InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: AddExtendedService**
Objective:	Register a QckExtendedService with the QckPlugin Framework.
Prototype:	`int AddExtendedService(const char* ServiceName, void* ServiceHandle);`
Parameters:	ServiceName — Name by which a service is to be registered with the QckPlugin service manager. ServiceHandle — Handle to the object that implements the service in question.
Return Values:	
Notes:	1. The AddExtendedService function allows a plug-in (QckPlugin or QckExtension) to register a service (QckExtended Service) with the QckPlugin Framework so that it can be used by other plug-ins in the system. 2. Typically this function is used by QckExtensions but it can also be used by a QckPlugin (that has some service to offer) as well. 3. This function must be called during the InitQckPlugin function call only. The QckExtended service added using this method will be available to other plug-ins only during the ExecuteQckPlugin function call. 4. The 'serviceName' must be a unique string that identifies the service to be added as well as it's API version (preferably). This string must be declared in the same header file that contains the service interface definition.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: GetExtendedService**
Objective:	Request access to a registered QckExtendedService.
Prototype:	`void* GetExtendedService(const char* ServiceName);`
Parameters:	ServiceName — Name of the service, access to which is desired.
Return Values:	a valid address — Handle to the QckExtended Service (success). null — The specfied service was not found.
Notes:	1. The GetExtendedService method allows a plug-in to access a QckExtended Service. 2. The 'ServiceName' of a QckExtended Service is declared in the header file corresponding to that service. 3. When this function is called, the QckPlugin Framework tries to locate the service corresponding to the name. If the service is not found, it returns NULL. 4. Since the order in which plug-ins (QckPlugins and QckExtensions) are loaded cannot be pre-determined, this function should not be used during the InitQckPlugin function call, unless specifically mentioned in the extended service usage.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckSrvcMgr :: GetNotificationHandler**	
Objective:	Get the address of a handler that handles a particular notification for a QckPlugin.	
Prototype:	`IQckEventNotify* GetNotificationHandler(void* pluginHandle, int notificationType);`	
Parameters:	pluginHandle	A handle that uniquely identifies a QckPlugin obtained via the InitQckPlugin function call.
	notificationType	The notification for which the handler being requested was set.
Return Values:	an address or NULL.	Address of the handler that recieves 'notificationType' notifications. NULL indicates that no handler was set to handle the specified notification.
Notes:		
Last Revision:	0x5653E56	
Sample Code:		

Class	**IQckWindowSrvc**	
Objective:	An interface to the QckWindow Service.	
Members:	none.	
Methods:	SetWindowServiceNotify	Enable/Disable notifications related to window QckEvents
	SelectPoint	Allow a user to select a point on Qckvu3 screen.
	SetDrawRubberBand	Set a mode to allow a user to select a rectangular window on the Qckvu3 screen.
	SetWindow	Set the view in the Qckvu3 viewing area to a particular window.
	GetCurrentWindow	Get the extents of the window currently set in the Qckvu3 viewing area.
	SelectPolygonArea	Allow a user to draw a polygonal area on the Qckvu3 screen.
Notes:	1.	
Last Revision:	0x5638C7F	

Method	**IQckWindowSrvc :: SetWindowServiceNotify**
Objective:	Enable/Disable notifications related to window QckEvents
Prototype:	`void SetWindowServiceNotify(void* plgHandle, IQckEventNotify* client);`
Parameters:	plgHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. client — Address of a valid QckEventHandler that handles notifications corresponding to window QckEvents.
Return Values:	none.
Notes:	1. The SetWindowServiceNotify method can be used to setup the notifications related to window QckEvents. 2. To start getting notifications, the 'client' parameter should be a handle to a valid QckEventHandler. To stop receiving notifications, it should be NULL. 3. When this method is called, the QckPlugin Framework records the handler in association with the plug-in handle. Only one handler is stored per plug-in for the QckWindow service. 4. When a QckEvent related to window operations occurs, the QckPlugin Framework finds all plug-ins with non-NULL handlers and calls the IQckEventNotify method (overridden by the handler) corresponding to that QckEvent. 5. This method can be utilized during both InitQckPlugin and ExecuteQckPlugin functions.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckWindowSrvc :: SelectPoint**
Objective:	Allow a user to select a point on Qckvu3 screen.
Prototype:	`void SelectPoint(void* plgHandle, IQckEventNotify* client);`
Parameters:	plgHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. client — Address of a valid QckEventHandler that handles the OnSelectPoint notification.
Return Values:	none.
Notes:	1. The SelectPoint method allows a user to select a point on Qckvu3 screen and notifies the plug-in about the co-ordinates of that point in user units (CAD data space). 2. The 'client' must be the address of a valid QckEventHandler that overrides the OnSelectPoint method. 3. When this method is called, the QckPlugin Framework records the plug-in handle and the notification handler. It then sets Qckvu3 in a mode (indicated by a change in the cursor icon) so that the user can pick a point with a mouse-click. 4. When the user selects a point, the QckPlugin Framework invokes the OnSelectPoint method defined by the 'client' and provides it the co-ordinates of the selected point. Immediately after that, Qckvu3 is set into the last operation mode before point selection (info, zoom, pan etc.). 5. Therefore, only one point selection can be done per call to this method. 6. If two or more plug-ins call this method before the point selection event occurs, then all those plug-ins will receive the co-ordinates of the selected point. 7. This method should be used only during the ExecuteQckPlugin function.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckWindowSrvc :: SetDrawRubberBand**
Objective:	Set a mode to allow a user to select a rectangular window on the Qckvu3 screen.
Prototype:	`void SetDrawRubberBand(bool onOff);`
Parameters:	onOff — true: Change cursor and set Qckvu3 to draw mode. false: Turn off draw mode and restore Qckvu3 to the previous mode (info, zoom, pan, measure etc.)
Return Values:	none.
Notes:	1. The SetDrawRubberBand method allows a user to select a rectangular window on Qckvu3 screen and notifies the plug-in about the window co-ordinates when the selection is done. 2. When this method is called with 'onOff' set to 'true', the QckPlugin Framework sets Qckvu3 to a mode (indicated by a cursor change) so that the user can draw a rectangular window on the Qckvu3 screen by click-drag-release of the left mouse button. 3. When the left mouse button is released, Qckvu3 searches for plug-ins that have specified a valid non-NULL QckEventHandler by means of the SetWindowServiceNotify method. 4. For each of those handlers, Qckvu3 calls the OnDrawWindowRubberBand method and passes the co-ordinates of the selected window in user units (CAD data space). 5. When this method is called with 'onOff' set to 'false', Qckvu3 is set into the last operation mode (zoom, pan, info etc.) before the window selection was activated. 6. If two or more plug-ins have specified a valid handler for the OnDrawWindowRubberBand method, all of them will be notified about the window selection even if SetDrawRubberBand was invoked by only one of them. 7. To stop receiving window selection notifications, a plug-in must call the SetWindowServiceNotify method with the 'client' parameter set to NULL. 8. This method should be used only during the ExecuteQckPlugin function.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckWindowSrvc :: SetWindow**
Objective:	Set the view in the Qckvu3 viewing area to a particular window.
Prototype:	`int SetWindow(const sQCKWINDOW& windowExts, bool refreshView = true);`
Parameters:	windowExts — Extents of the window (in user units) to be set on the Qckvu3 screen. refreshView — true: Refresh the view on the screen (invoke a redraw) false: Change the internal database but do not cause a redraw.
Return Values:	
Notes:	1. The SetWindow sets the Qckvu3 screen (and view) to a rectangular window into the CAW data space (user units). 2. If 'refreshView' parameter is set to 'true', the Qckvu3 screen is updated immediately to reflect the new window (by doing a redraw). 3. If 'refreshView' parameter is set to 'off', this method changes the Qckvu3 view (database) but does not update the screen. This mode is useful when SetWindow is used in conjunction with SetViewCell and/or one of more SetLayerOnOff methods, to avoid flashes as Qckvu3 tries to redraw the screen for each of those methods. In such a case, the plug-in can simply call Redraw at the end to reflect all the changes in one refresh. 4. This method should be used only during the ExecuteQckPlugin function.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckWindowSrvc :: GetCurrentWindow**
Objective:	Get the extents of the window currently set in the Qckvu3 viewing area.
Prototype:	`void GetCurrentWindow(sQCKWINDOW& windowExts);`
Parameters:	windowExts — A buffer to retreive the extents (in user units) of the current view window.
Return Values:	none.
Notes:	1. This method should be used only during the ExecuteQckPlugin function.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckWindowSrvc :: SelectPolygonArea**
Objective:	Allow a user to draw a polygonal area on the Qckvu3 screen.
Prototype:	`void SelectPolygonArea(void* pluginHandle, IQckEventNotify* eventHandler);`
Parameters:	pluginHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. eventHandler — Address of a valid QckEventHandler that handles the OnSelectPolygonArea notification.
Return Values:	
Notes:	1. The SelectPolygonArea method allows a user to select a polygonal window on the Qckvu3 screen through a series of mouse clicks (single-click) and notifies the plug-in about the co-ordinates of the polygonal window when the selection is complete (double-click). 2. The 'eventHandler' should be the address of a valid QckEventHandler that overrides the OnSelectPolygonArea method. 3. When this method is called, Qckvu3 is set to a mode (indicated by cursor change) so that the user can draw a polygonal window on Qckvu3 screen. When the user completes the selection with a double-click, Qckvu3 invokes the OnSelectPolygonArea method of the handler and then restores Qckvu3 in the last operation mode (zoom, info, pan etc.) before this method was called. 4. Therefore, only one polygon window can be selected per call to this method. 5. However, if more than one plug-in call this method in succession, before the user makes a selection, each of those plug-ins will be notified of the selected polygonal window once the selection is made.
Last Revision:	0x5638C7F
Sample Code:	

Class	**IQckMenuSrvc**	
Objective:	An interface to the QckMenu Service.	
Members:	none.	
Methods:	CreateMenuItem	Create an entry into the Qckvu3 'Tools' menu associated with a QckPlugin.
	CreatePopupMenu	Create a new menu on the Qckvu3 menu bar associated with a QckPlugin.
	AddPopupMenuItem	Add an entry to a menu created by a QckPlugin.
	GetAppShell	Get Qckvu3 main shell widget (UNIX/LINUX only).
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckMenuSrvc :: CreateMenuItem**
Objective:	Create an entry into the Qckvu3 'Tools' menu associated with a QckPlugin.
Prototype:	`int CreateMenuItem(void* pluginHandle, const char* itemName);`
Parameters:	pluginHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. itemName — Name of the menu item as it should appear in the 'Tools' menu.
Return Values:	
Notes:	1. The CreateMenuItem method is used by QckPlugins to add a menu item to the Qckvu3 'Tools' menu so that the Qckvu3 user may invoke the QckPlugin using Qckvu3 user interface. 2. When this method is called, Qckvu3 creates a new menu item in Qckvu3 'Tools' menu and associates it with the QckPlugin. 3. A single QckPlugin can add multiple menu-items so that it can house multiple features which can be invoked directly from Qckvu3 menu. 4. If successful, this method returns a unique number (ID) that identifies the menu-item added from other menu-items on the Qckvu3 menu bar. 5. When the user clicks on the menu item, Qckvu3 calls the ExecuteQckPlugin defined inside the corresponding QckPlugin, therefore transferring control to that QckPlugin. 6. As a parameter to the ExecuteQckPlugin function, it also passes the ID of the menu-item that was clicked by the user. This way, the QckPlugin knows which feature is requested and can take appropriate action. Therefore it is advisable to remember the return value for later use during the ExecuteQckPlugin function. 7. This function must be called only during the InitQckPlugin function call.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckMenuSrvc :: CreatePopupMenu**
Objective:	Create a new menu on the Qckvu3 menu bar associated with a QckPlugin.
Prototype:	`void* CreatePopupMenu(void* pluginHandle, const char* itemName, bool newMenu = false);`
Parameters:	pluginHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. itemName — Name of the popup menu as it should appear to the user. newMenu — true: Create a new menu on Qckvu3 Menu Bar. false: Add a new sub-menu to the Qckvu3 'Tools' menu.
Return Values:	
Notes:	1. Since a QckPlugin can house multiple features, it would be convinient and desirable for a QckPlugin to group the menu-items corresponding to these 'features' into one or more popup menus. 2. The CreatePopupMenu method creates an empty menu either as a child of the 'Tools' menu or as a new menu on the Qckvu3 menu bar depending on the 'newMenu' parameter. 3. If successful, this method returns a handle to the newly created menu. 4. This handle can be used to add individual menu-items to the newly created menu using the AddPopupMenuItem method. 5. All children (menu items) of a popup menu are associated with the same QckPlugin as the popup menu itself. 6. This function must be called only during the InitQckPlugin function call.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckMenuSrvc :: AddPopupMenuItem**
Objective:	Add an entry to a menu created by a QckPlugin.
Prototype:	`int AddPopupMenuItem(void* popupMenuHandle, const char* itemName);`
Parameters:	popupMenuHandle — Handle of the menu under which this menu item will be listed. itemName — Name of the menu item to be added as it should appear to the user.
Return Values:	
Notes:	1. The AddPopupMenuItem method is used to add menu-items to a popup menu created using the CreatePopupMenu method. 2. Every child menu-item added to a popup menu is associated with the same QckPlugin as the popup menu itself. 3. If successful, this method returns a number (ID) that uniquely identifies the newly added menu-item from others in the Qckvu3 menu. 4. When the user clicks on a specific menu-item, Qckvu3 identifies the QckPlugin associated with the parent popup menu and calls the ExecuteQckPlugin function defined inside that QckPlugin. 5. Further, it passes the ID of the menu-item that was clicked, as a parameter to the ExecuteQckPlugin function so that the QckPlugin can take appropriate action. Therefore it is advisable to remember the return value for later use during the ExecuteQckPlugin function. 6. This method must be used only during the InitQckPlugin function call.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckMenuSrvc :: GetAppShell (UNIX/Linux Only)**
Objective:	Get Qckvu3 main shell widget (UNIX/LINUX only).
Prototype:	`void* GetAppShell();`
Parameters:	none.
Return Values:	A valid address. Handle (type 'Widget') to Qckvu3 application shell (success)
Notes:	1. The GetAppShell method is meant to be used by QckPlugins built with Motif library (Xm). 2. The method returns a widget handle which represents the Qckvu3 application shell. This can be used as parent for creating the motif dialogs inside the QckPlugin. 3. This method can be called during the InitQckPlugin or ExecuteQckPlugin function calls.
Last Revision:	0x5622522
Sample Code:	

Class	**IQckMarkerSrvc**	
Objective:	An interface to the QckMarkerService.	
Members:	none.	
Methods:	DrawMarker	Draw a marker at a specific location on Qckvu3 screen.
	ZoomToMarker	Zoom to a specific marker on Qckvu3 screen.
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckMarkerSrvc :: DrawMarker**
Objective:	Draw a marker at a specific location on Qckvu3 screen.
Prototype:	`int DrawMarker(const sQCKMARKER& marker);`
Parameters:	marker data describing the marker to be drawn.
Return Values:	
Notes:	1. This method is used to place a marker somewhere in the CAD data space. 2. This method should be used only during the ExecuteQckPlugin function. 3. It is the plug-in's responsibility to store and manage all the markers. Qckvu3 does not keep any information about the markers. 4. Therefore this method does not draw a persistant object. The marker drawn by using this method will disappear when the screen is refreshed (Redraw). 5. Therefore, the OnRedraw notification must be captured and this method should be called again inside the OnRedraw notification.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckMarkerSrvc :: ZoomToMarker**
Objective:	Zoom to a specific marker on Qckvu3 screen.
Prototype:	`int ZoomToMarker(const sQCKMARKER& marker, double windowSizeX, double windowSizeY, bool refresh = true);`
Parameters:	marker — data describing the marker. windowSizeX — width of the window in pixels centered at the marker. windowSizeY — height of the window in pixels centered at the marker. refresh — true: invoke a redraw. false: do not invoke a redraw.
Return Values:	
Notes:	1. The ZoomToMarker is a convinience method based on SetWindow that changes the Qckvu3 view window such that it is centered around a specific marker. 2. If 'refresh' is set to true, a Redraw will be invoked immediately after the window is changed to refresh the screen. 3. Note that the ZoomToMarker method does not draw the marker. It simply changes the view window. It is the plug-in's responsibility to draw the marker if needed either by calling DrawMarker immediately after or by setting 'refresh' to true and calling DrawMarker inside the OnRedraw notification.
Last Revision:	0x5622522
Sample Code:	

Class	**IQckExtractSrvc**	
Objective:	An interface to the QckExtractService.	
Members:	none.	
Methods:	ExtractGDSII	Extract a rectangular window from the file loaded in Qckvu3 to a GDSII file.
	ExtractGDSII	Extract a polygonal window from the file loaded in Qckvu3 to a GDSII file.
Notes:	1.	
Last Revision:	0x5638C7F	

Method	**IQckExtractSrvc :: ExtractGDSII (Rectangular Window based)**
Objective:	Extract a rectangular window from the file loaded in Qckvu3 to a GDSII file.
Prototype:	`int ExtractGDSII(const sQCKEXTRACTWINDOW& params);`
Parameters:	params Extraction parameters.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckExtractSrvc :: ExtractGDSII (Polygonal Winow based)**
Objective:	Extract a polygonal window from the file loaded in Qckvu3 to a GDSII file.
Prototype:	`int ExtractGDSII(const sQCKEXTRACTPOLY& params);`
Parameters:	params Extraction parameters.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Class	**IQckGraphicsSrvc**	
Objective:	An interface to the QckGraphicsService.	
Members:	none.	
Methods:	SetDrawServiceNotify	Enable/Disable notifications for QckEvents related to the Graphics QckService.
	Redraw	Refresh the view screen.
	SelectForegroundColor	Select the default foreground color to draw objects on the Qckvu3 screen.
	SelectColor	Select a particular RGB color to draw objects on the Qckvu3 screen.
	DrawBoundary	Draw a boundary on the Qckvu3 screen.
	IsSmallObject	
	SetLineSize	Specify the width of the line to draw objects on Qckvu3 screen.
	GetPixelSize	Get the size of a screen pixel in user units based on the current zoom level.
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckGraphicsSrvc :: SetDrawServiceNotify**
Objective:	Enable/Disable notifications for QckEvents related to the Graphics QckService.
Prototype:	`void SetDrawServiceNotify(void* plgHandle, IQckEventNotify* client);`
Parameters:	plgHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. client — Address of a valid QckEventHandler that handles the OnRedraw notification.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckGraphicsSrvc :: Redraw**	
Objective:	Refresh the view screen.	
Prototype:	`void Redraw(const bool forceredraw, bool zoomHome = false);`	
Parameters:	forcedraw	Redresh the screen using the cached image (fast) or perform a complete redraw (slower).
	zoomHome	Redraw the home view.
Return Values:		
Notes:	1.	
Last Revision:	0x5653E56	
Sample Code:		

Method	**IQckGraphicsSrvc :: SelectForegroundColor**
Objective:	Select the default foreground color to draw objects on the Qckvu3 screen.
Prototype:	`void SelectForegroundColor();`
Parameters:	none.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckGraphicsSrvc :: SelectColor**
Objective:	Select a particular RGB color to draw objects on the Qckvu3 screen.
Prototype:	`bool SelectColor(unsigned short R, unsigned short G, unsigned short B);`
Parameters:	R,G,B The RGB color values (0-255) to be used to draw objects using the IQckGraphicsSrvc::DrawBoundary method.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckGraphicsSrvc :: DrawBoundary**	
Objective:	Draw a boundary on the Qckvu3 screen.	
Prototype:	`int DrawBoundary(const int type,const double *userUnits,const int numVert,const bool drawOutline);`	
Parameters:	type	type of the object to be drawn. (eQckDrawBoundaryType)
	userUnits	X,Y Co-ordinates (in user units) of the vertices forming the object.
	numVert	Number of vertices in the object. (e.g rectangle has 5 vertices)
	drawOutline	
Return Values:		
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckGraphicsSrvc :: IsSmallObject**	
Objective:		
Prototype:	`bool IsSmallObject(int numVertices, const double* xyCoords, int thresholdInPixels, sQCKWINDOW& newExtentsBox);`	
Parameters:	numVertices	Number of vertices in the object. (e.g rectangle has 5 vertices)
	xyCoords	X,Y Co-ordinates (in user units) of the vertices forming the object.
	thresholdInPixels	Threshold size to be used when deciding if an object is too small to be drawn in its entirety.
	newExtentsBox	The extents of the box used to represent the object if it is considered too small to be drawn in its entirety.
Return Values:		
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckGraphicsSrvc :: SetLineSize**	
Objective:	Specify the width of the line to draw objects on Qckvu3 screen.	
Prototype:	`void SetLineSize(unsigned int lineSize);`	
Parameters:	lineSize	Specify the thickness (in pixels) to be used when drawing a boundary using the IQckGraphicsSrvc::DrawBoundary method.
Return Values:		
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckGraphicsSrvc :: GetPixelSize**
Objective:	Get the size of a screen pixel in user units based on the current zoom level.
Prototype:	`double GetPixelSize();`
Parameters:	none.
Return Values:	The size of a pixel in user units.
Notes:	1.
Last Revision:	0x5638C7F
Sample Code:	

Class	**IQckQuerySrvc**	
Objective:	An interface to the QckQueryService.	
Members:	none.	
Methods:	Stop	Stop the execution of a query.
	SetQueryServiceNotify	Enable/Disable notifications for QckEvents related to the Query QckService.
	SetViewerInfoMode	Set Qckvu3 to allow a user to pick an object from the screen.
	SetReferenceVectorMatrix	Control the format in which transformation information is returned for cell references during the execution of a query.
	GetDataVector	Start a new query based on the current view cell, layer window and other settings.
	GetCellReferences	Get the instances where a particular cell is rerenced based on the current view cell, layer, window and other settings.
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckQuerySrvc :: Stop**
Objective:	Stop the execution of a query.
Prototype:	`void Stop();`
Parameters:	none.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckQuerySrvc :: SetQueryServiceNotify**
Objective:	Enable/Disable notifications for QckEvents related to the Query QckService.
Prototype:	`oid SetQueryServiceNotify(void* plgHandle, IQckEventNotify* client);`
Parameters:	plgHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. client — Address of a valid QckEventHandler that handles notifications related to the QckQuery Service.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckQuerySrvc :: SetViewerInfoMode**
Objective:	Set Qckvu3 to allow a user to pick an object from the screen.
Prototype:	`void SetViewerInfoMode(bool setOnOff);`
Parameters:	setOnOff — true: Set Qckvu3 into info mode. false: Set Qckvu3 into the previous mode. (zoom, pan, info, measure etc.)
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckQuerySrvc :: SetReferenceVectorMatrix**
Objective:	Control the format in which transformation information is returned for cell references during the execution of a query.
Prototype:	`void SetReferenceVectorMatrix(bool setOnOff);`
Parameters:	setOnOff — true: Get scale, rotation and reflection information (for cell and array references (sQCKSREF, sQCKAREF)) as a 2x2 transformation matrix of type sTMATRIX. false: Get scale, rotation and reflection information (for cell and array references (sQCKSREF, sQCKAREF)) as members of a struct of type sTPARAMS.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckQuerySrvc :: GetDataVector**
Objective:	Start a new query based on the current view cell, layer window and other settings.
Prototype:	`int GetDataVector(IQckEventNotify* client, const sQCKQUERYSETTINGS* settings = 0);`
Parameters:	client — Address of a valid QckEventHandler that handles the OnGetDataVector notification. settings — Localized parameters to customize the query.
Return Values:	
Notes:	1.
Last Revision:	0x56238E4
Sample Code:	

Method	**IQckQuerySrvc :: GetCellReferences**
Objective:	Get the instances where a particular cell is rerenced based on the current view cell, layer, window and other settings.
Prototype:	`int GetCellReferences(const char* cellName, int nestingLevel, IQckEventNotify* client);`
Parameters:	cellName — Name of the cell whose references are to be obtained. nestingLevel — Return references that lie within the specified nesting level only. client — Address of a valid QckEventHandler that handles the OnGetDataVector notification.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Class	**IQckDatabaseSrvc**	
Objective:	An interface to the QckDatabaseService.	
Members:	none.	
Methods:	SetDatabaseServiceNotify	Enable/Disable notifications corresponding to database QckEvents.
	SetViewCell	Change the current view cell.
	GetGrid	Get grid information for the file loaded into Qckvu3.
	GetUnits	Get units information for the file loaded into Qckvu3.
	GetCellID	Get an unique index corresponding to a cell name.
	GetCellName	Get the cell name for a valid cell index.
	GetViewCell	Get the name of the cell currently opened for viewing.
	GetCellList	Get list of cells present in the file loaded into Qckvu3.
	GetCellRoot	Get the list of top cells present in the file loaded into Qckvu3.
	GetCellParents	Get list of cells (parents) that reference a particular cell (child).
	GetCellChildren	Get a list of cells (children) that are referenced by a particular cell (parent).
	GetCellExtents	Get the extents in user units for a particular cell.
	GetCellVertices	Get the number of boundary and path vertices in a cell.
	GetCellInfo	Get the number of boundaries, paths, cell references, array references and texts in a cell.
	GetLayerList (as shorts)	Get a complete list of layers and datatypes present in the file loaded in Qckvu3 as two lists of short integers.
	GetLayerList (as strings)	Get a complete list of layers and datatypes present in the file loaded in Qckvu3 as a list of strings formatted as "layer:datatype".
	GetDatatypesForLayer	Get a list of datatypes present for a particular layer number.
	FreeCellList	Free the memory allocated by the GetCellList, GetCellRoot, GetCellChildren and GetCellParents mentods.
	FreeLayerList (shorts)	Free the memory associated with the GetLayerList and GetDatatypesForLayer methods.
	FreeLayerList (strings)	Free the memory associated with the GetLayerList method.
Notes:	1.	
Last Revision:	0x5638C7F	

Method	**IQckDatabaseSrvc :: SetDatabaseServiceNotify**
Objective:	Enable/Disable notifications corresponding to database QckEvents.
Prototype:	`void SetDatabaseServiceNotify(void* plgHandle, IQckEventNotify* client);`
Parameters:	plgHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. client — Address of a valid QckEventHandler that handles notifications related to the QckDatabase Service.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: SetViewCell**
Objective:	Change the current view cell.
Prototype:	`int SetViewCell(const char* cellName, bool refreshView = true);`
Parameters:	cellName — Name of the cell to be opened. refreshView — true: Redraw Qckvu3 screen after the new cell is opened. false: Do not change the current Qckvu3 view. (wait for an explicit call to Redraw)
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetGrid**
Objective:	Get grid information for the file loaded into Qckvu3.
Prototype:	`int GetGrid(double& aBuffer);`
Parameters:	aBuffer A buffer to store the grid value.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetUnits**
Objective:	Get units information for the file loaded into Qckvu3.
Prototype:	`int GetUnits(double& aBuffer, char* aStrBuffer = 0);`
Parameters:	aBuffer A buffer to store the units as a value. aStrBuffer Address of a buffer to store units as a string (char string).
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetCellID**
Objective:	Get an unique index corresponding to a cell name.
Prototype:	`int GetCellID(const char* aCellName);`
Parameters:	aCellName Name of a cell present in the file loaded into Qckvu3.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetCellName**
Objective:	Get the cell name for a valid cell index.
Prototype:	`const char* GetCellName(int anID);`
Parameters:	anID A valid ID obtained using the GetCellID method.
Return Values:	char string Name of the cell corresponding to the specified ID. (success) null Invalid ID. (failure)
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetViewCell**	
Objective:	Get the name of the cell currently opened for viewing.	
Prototype:	`const char* GetViewCell();`	
Parameters:	none.	
Return Values:	char string null	Name of the cell corresponding to the specified ID. (success) Invalid ID. (failure)
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetCellList**	
Objective:	Get list of cells present in the file loaded into Qckvu3.	
Prototype:	`int GetCellList(char*** aCellListPtr);`	
Parameters:	aCellListPtr	Adress of a pointer which will point to a list of cell names if this method returns successfully.
Return Values:	return > 0	Number of cells present in the list pointed by 'aCellListPtr' (success)
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetCellRoot**	
Objective:	Get the list of top cells present in the file loaded into Qckvu3.	
Prototype:	`int GetCellRoot(char*** aCellListPtr);`	
Parameters:	aCellListPtr	Adress of a pointer which will point to a list of cell names if this method returns successfully.
Return Values:	return > 0	Number of cells present in the list pointed by 'aCellListPtr' (success)
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetCellParents**	
Objective:	Get list of cells (parents) that reference a particular cell (child).	
Prototype:	`int GetCellParents(const char* aCell, char*** aCellListPtr);`	
Parameters:	aCell	Name of a cell present in the file loaded into Qckvu3.
	aCellListPtr	Adress of a pointer which will point to a list of cell names if this method returns successfully.
Return Values:	return >= 0	Number of cells present in the list pointed by 'aCellListPtr' (success).
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetCellChildren**	
Objective:	Get a list of cells (children) that are referenced by a particular cell (parent).	
Prototype:	`int GetCellChildren(const char* aCell, char*** aCellListPtr);`	
Parameters:	aCell	Name of a cell present in the file loaded in Qckvu3.
	aCellListPtr	Address of a pointer that will point to a list of cell names when this method returns succesfully.
Return Values:	return >= 0	Number of cells present in the list pointed by 'aCellListPtr' (success).
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetCellExtents**
Objective:	Get the extents in user units for a particular cell.
Prototype:	`int GetCellExtents(const char* aCell, sQCKWINDOW& aBuffer);`
Parameters:	aCell Name of a cell present in the file loaded in Qckvu3. aBuffer Buffer to store the extents box (in user units) of the specified cell.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetCellVertices**
Objective:	Get the number of boundary and path vertices in a cell.
Prototype:	`int GetCellVertices(const char* aCell, LongLong_t& nBVertices, LongLong_t& nPVertices);`
Parameters:	aCell — Name of a cell present in the file loaded in Qckvu3. nBVertices — Buffer to store number of boundary vertices in the specified cell. nPVertices — Buffer to store number of path vertices in the specified cell.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetCellInfo**
Objective:	Get the number of boundaries, paths, cell references, array references and texts in a cell.
Prototype:	`int GetCellInfo(const char* aCell, LongLong_t& nBndrys, LongLong_t& nPaths, LongLong_t& nTexts, LongLong_t& nSrefs, LongLong_t& nArefs);`
Parameters:	aCell — Name of a cell present in the file loaded in Qckvu3. nBndrys — Buffer to store number of boundaries in the specified cell. nPaths — Buffer to store number of paths in the specified cell. nTexts — Buffer to store number of texts in the specified cell. nSrefs — Buffer to store number of cell references in the specified cell. nArefs — Buffer to store number of array references in the specified cell.
Return Values:	
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: GetLayerList (as shorts)**	
Objective:	Get a complete list of layers and datatypes present in the file loaded in Qckvu3 as two lists of short integers.	
Prototype:	`int GetLayerList(unsigned short** layerListPtr, unsigned short** dtListPtr);`	
Parameters:	layerListPtr	Address of a pointer that will point to a list of layer numbers if this method returns successfully.
	dtListPtr	Address of a pointer that will point to a list of data-type numbers if this method returns successfully.
Return Values:	return > 0	Number of layers present in the list pointed by 'layerListPtr'
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetLayerList (as strings)**	
Objective:	Get a complete list of layers and datatypes present in the file loaded in Qckvu3 as a list of strings formatted as "layer:datatype".	
Prototype:	`int GetLayerList(char*** stringListPtr);`	
Parameters:	stringListPtr	The address of a pointer (char**) which will point to a list of strings if this function returns successfully.
Return Values:	return > 0	Number of layers present in the list pointed by 'layerListPtr'
Notes:	1.	
Last Revision:	0x5638C7F	
Sample Code:		

Method	**IQckDatabaseSrvc :: GetDatatypesForLayer**
Objective:	Get a list of datatypes present for a particular layer number.
Prototype:	`int GetDatatypesForLayer(unsigned short aLayer, unsigned short** dtListPtr);`
Parameters:	aLayer — A valid layer number for which a list of datatypes is to be obtained. dtListPtr — Address of a pointer that will point to a list of datatypes if this method returns successfully.
Return Values:	return > 0 — Number of items (datatypes) present in the list pointed by 'dtListPtr'
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: FreeCellList**
Objective:	Free the memory allocated by the GetCellList, GetCellRoot, GetCellChildren and GetCellParents methods.
Prototype:	`void FreeCellList(char** aList, int nListItems);`
Parameters:	aList — List of cell names to be de-allocated. nListItems — Number of items (cell names) present in the list.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: FreeLayerList (shorts)**
Objective:	Free the memory associated with the GetLayerList and GetDatatypesForLayer methods.
Prototype:	`static void FreeLayerList(unsigned short* layerList);`
Parameters:	layerList List of layers or datatypes to be de-allocated.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckDatabaseSrvc :: FreeLayerList (strings)**
Objective:	Free the memory associated with the GetLayerList method.
Prototype:	`static void FreeLayerList(char** layerList, int numItems);`
Parameters:	layerList List of layers,datatypes strings to be de-allocated. numItems Number of strings present in the list.
Return Values:	none.
Notes:	2.
Last Revision:	0x5638C7F
Sample Code:	

Class	**IQckLayerSrvc**	
Objective:	An interface to the QckLayerService.	
Members:	none.	
Methods:	SetLayerServiceNotify	Enable/Disable notifications corrsponding to layer QckEvents.
	SetLayerOnOff	Turn a specific layer on or off.
	SetAllLayersOnOff	Turn all layers on or off.
	GetLayerOnOff	Get the on/off state for a particular layer.
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckLayerSrvc :: SetLayerServiceNotify**
Objective:	Enable/Disable notifications corrsponding to layer QckEvents.
Prototype:	`void SetLayerServiceNotify(void* plgHandle, IQckEventNotify* client);`
Parameters:	plgHandle — The plug-in handle obtained from the parameter list of the InitQckPlugin function. client — Address of a valid QckEventHandler that handles notifications related to the QckLayer Service.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckLayerSrvc :: SetLayerOnOff**
Objective:	Turn a specific layer on or off.
Prototype:	`void SetLayerOnOff(unsigned short layer, unsigned short dataType, bool onOff = true, bool refresh = true);`
Parameters:	layer — Layer number of the layer to be toggled. datatype — Datatype of the layer to be toggled. onOff — true: Turn the specified layer ON. false: Turn the specified layer OFF. refresh — true: Refresh the Qckvu3 screen (implicit Redraw). false: Do not refresh the Qckvu3 screen (wait for an explicit Redraw).
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckLayerSrvc :: SetAllLayersOnOff**
Objective:	Turn all layers on or off.
Prototype:	`void SetAllLayersOnOff(bool onOff = true, bool refresh = true);`
Parameters:	onOff — true: Turn ON all layers. false: Turn OFF all layers. refresh — true: Refresh the Qckvu3 screen (implicit Redraw). false: Do not refresh the Qckvu3 screen (wait for an explicit Redraw).
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckLayerSrvc :: GetLayerOnOff**	
Objective:	Get the on/off state for a particular layer.	
Prototype:	`bool GetLayerOnOff(unsigned short layer, unsigned short dataType);`	
Parameters:	layer dataType	Layer number of the layer whose ON/OFF state is to be determined. Datatype number of the layer whose ON/OFF state is to be determined.
Return Values:	true false	The specified layer is visible (ON). The specified layer is invisible (OFF).
Notes:	1.	
Last Revision:	0x5622522	
Sample Code:		

Class	**IQckEventNotify**	
Objective:	A base class for a QckEvent notification handler. To receive a particular notification, the QckPlugin notification handler must over-ride the virtual method corresponding to that notification.	
Members:	none.	
Methods:	OnDrawWindowRubberBand	Invoked when a user draws a window on Qckvu3 screen.
	OnChangeViewWindow	Invoked when the Qckvu3 view window changes.
	OnSelectPoint	Invoked when a user selects a point of Qckvu3 screen after it was set into the 'point selection mode' by means of the IQckWindowSrvc::SelectPoint method.
	OnRedraw	Invoked when Qckvu3 has refreshed the screen.
	OnGetDataVector	Invoked when the Qckvu3 query finds a vector data item that satisfies the query parameters.
	OnInfoObject	Invoked when the user finds an object from Qckvu3 screen using the info tool.
	PreFileOpen	Invoked just before a file is about to be opened.
	PostFileOpen	Invoked just after a file is loaded into Qckvu3.
	PreCellOpen	Invoked just before a cell is about to be opened into the viewer.
	PostCellOpen	Invoked just after a cell is opened in Qckvu3.
	OnSelectPolygonArea	Invoked when the user draws a polygonal window on Qckvu3 screen.
	OnChangeLayers	Invoked when the layer settings have changed.
	OnShowDialog	
Notes:	1.	
Last Revision:	0x5622522	

Method	**IQckEventNotify :: OnDrawWindowRubberBand**
Objective:	Invoked when a user draws a window on Qckvu3 screen.
Prototype:	`void OnDrawWindowRubberBand(const sQCKWINDOW& windowExts);`
Parameters:	windowExts — Extents (in user units) of the window drawn by the user on Qckvu3 screen.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnChangeViewWindow**
Objective:	Invoked when the Qckvu3 view window changes.
Prototype:	`void OnChangeViewWindow(const sQCKWINDOW& windowExts);`
Parameters:	windowExts — Extents (in user units) of the new view window set on Qckvu3 screen.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnSelectPoint**
Objective:	Invoked when a user selects a point of Qckvu3 screen after it was set into the 'point selection mode' by means of the IQckWindowSrvc::SelectPoint method.
Prototype:	`void OnSelectPoint(double x, double y);`
Parameters:	x,y The X and Y co-ordinates (in user units) of the point selected by the user on Qckvu3 screen.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnRedraw**
Objective:	Invoked when Qckvu3 has refreshed the screen.
Prototype:	`void OnRedraw();`
Parameters:	none.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnGetDataVector**
Objective:	Invoked when the Qckvu3 query finds a vector data item that satisfies the query parameters.
Prototype:	`void OnGetDataVector(const sQCKVECTOR& vectorDatum);`
Parameters:	vectorDatum — A single vector data item (boundary, path, text, cell reference or array reference) that satisfies the conditions of the query invoked by one of the IQckQuerySrvc methods.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnInfoObject**
Objective:	Invoked when the user finds an object from Qckvu3 screen using the info tool.
Prototype:	`void OnInfoObject(const sQCKINFOVECTOR& vectorDatum);`
Parameters:	vectorDatum — The single vector data item (boundary, path, text, cell reference or array reference) picked by the user on Qckvu3 screen.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: PreFileOpen**
Objective:	Invoked just before a file is about to be opened.
Prototype:	`int PreFileOpen();`
Parameters:	none.
Return Values:	0 Inform Qckvu3 to continue with file open. non-zero Inform Qckvu3 to abort file open.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: PostFileOpen**
Objective:	Invoked just after a file is loaded into Qckvu3.
Prototype:	`void PostFileOpen(const char* openedFileName, int fileType);`
Parameters:	openedFileName — Complete path of the file newly loaded into Qckvu3. fileType — Type of file loaded in Qckvu3. (eQckFileType)
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: PreCellOpen**
Objective:	Invoked just before a cell is about to be opened into the viewer.
Prototype:	`int PreCellOpen();`
Parameters:	none.
Return Values:	0 Inform Qckvu3 to continue cell open. non-zero Inform Qckvu3 to abort cell open.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: PostCellOpen**
Objective:	Invoked just after a cell is opened in Qckvu3.
Prototype:	`void PostCellOpen(const char* openedCellName);`
Parameters:	openedCellName Name of the newly loaded cell.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnSelectPolygonArea**
Objective:	Invoked when the user draws a polygonal window on Qckvu3 screen.
Prototype:	`void OnSelectPolygonArea(const sQCKPOLYWINDOW& polyExts);`
Parameters:	polyExts X,Y co-ordinates of the polygon drawn by the user on Qckvu3 screen.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnChangeLayers**
Objective:	Invoked when the layer settings have changed.
Prototype:	`void OnChangeLayers(bool showDatatypes);`
Parameters:	showDatatypes — true: Show Datatypes control is ON, data types are visible. false: Show Datatypes control is OFF, data types are not visible.
Return Values:	none.
Notes:	1.
Last Revision:	0x5622522
Sample Code:	

Method	**IQckEventNotify :: OnShowDialog (WINDOWS only)**
Objective:	
Prototype:	`void OnShowDialog(bool show);`
Parameters:	
Return Values:	none.
Notes:	1.
Last Revision:	0x5638C7F
Sample Code:	

Struct	**sQCKPLUGINARGS**
Objective:	Represents a single item of the array of arguments that Qckvu3 provides to a QckPlugin when it calls the ExecuteQckPlugin function.
Members:	mType Indicates what kind of data the argument 'mArg' represents. mArg Contains the argument data.
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sQCKWINDOW**	
Objective:	Represents a rectangular window (in user units) into the CAD data space.	
Members:	mMinX, mMinY, mMaxX, mMaxY	Co-ordinate values of the lower left and upper right vertices of the rectangular window.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKPOLYWINDOW**
Objective:	Represents a polygonal window (in user units) into the CAD data space.
Members:	mNumVert — Number of vertices forming a closed polygon. (e.g a square would have 5 vertices) mXY — Array of X,Y co-ordinate values of the polygonal window.
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sQCKMARKER**
Objective:	Represents a marker that can be used to mark a point or an object visually on Qckvu3 screen.
Members:	mShape — An identifier that represents the marker shape. (eMarkerShapeType) mRGB — The RGB color values (0-255) to used for rendering the marker. mSize — The marker size in pixels. mThickness — Thickness (in pixels) of the lines forming the marker.
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sQCKEXTRACTPARAMS**	
Objective:	Represents a set of basic parameters to control the output of a GDSII extraction.	
Members:	mOutputFileName	Name of the output GDSII file to which data will be extracted.
	mTopCellName	Name of the cell to which the extracted data will belong.
	mClip	true: Clip data along the extraction window. false: Extract any polygon crossing the extraction window without clipping.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKEXTRACTWINDOW**	
Objective:	Represents a set of parameters to control a GDSII extraction from a rectangular window.	
Members:	mWindow	Extents of the rectangular window of interest to be used for extraction.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKEXTRACTPOLY**
Objective:	Represents a set of parameters to control a GDSII extraction from a polygonal window.
Members:	mPoly Extents of the polygonal window of interest to be used for extraction.
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sQCKBOUNDARY**	
Objective:	Represents a single boundary in the CAD data space. (user units)	
Members:	mCellName	Name of the cell to which the boundary belongs.
	mLayer	Layer number of the layer on which the boundary is present.
	mDataType	Datatype of the layer on which the boundary is present.
	mNumVert	Number of vertices forming the closed boundary. (A rectangle has 5 vertices)
	mXY	Array of X,Y co-ordinates of the boundary vertices. (contains mNumVert*2 elements of type 'double')
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKPATH**	
Objective:	Represents a single path in the CAD data space. (user units)	
Members:	mType	Type of path (R: Round path, F: Flush Path, H: Half-extended Path)
	mCellName	Name of the cell to which the path belongs.
	mLayer	Layer number of the layer on which the path is present.
	mDataType	Datatype of the layer on which the path is present.
	mNumVert	Number of vertices forming the path.
	mXY	Array of X,Y co-ordinates of the path vertices. (contains mNumVert*2 elements of type 'double')
	mWidth	Width of the path in user units.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKSREF**
Objective:	Represents a single instance of a cell reference in the CAD data space. (user units)
Members:	mRefName — Name of the cell being referenced. mParentName — Name of the parent cell to which this reference belongs. mInsertion — X,Y co-odinates (in user units) of the point where the referenced cell is inserted. mExtentsXY — Extents box of this cell reference. (in user units). mMatrix — true: Transformation parameters are represented as a 2x2 transformation matrix (mTMatrix). false: Transformation parameters are represented as scale, rotation and reflection (mTParams). mTMatrix — Transformation parameters are represented as a 2x2 transformation matrix. mTParams — Transformation parameters are represented as scale, rotation and reflection.
Methods:	none.
Notes:	1.
Last Revision:	0x56559B6

Struct	**sPARENTREF**
Objective:	Represents a bounding box (extents box) of a single instance of a cell reference in the CAD data space. (user units)
Members:	mCellID — ID of the cell being referenced. (used with GetCellID and GetCellName) mExtents — Extents box of this cell reference. (in user units).
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sQCKHSREF**	
Objective:	Represents a single instance of a cell reference in the CAD data space just like sQCKSREF along with extents information about its parents and grandparents all the way to the TOP cell.	
Members:	mNumParents	Number of parents and grandparents referencing this instance of a cell reference all the way to the topmost cell.
	mParentTree	Array of extents information corrsponding to each parent or grandparent. (contains mNumParents elements).
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sTMATRIX**
Objective:	Represents the transformation parameters (scale, rotation, reflection) specific to a cell reference or an array reference, in form of a 2x2 transformation matrix.
Members:	mTm — A 2x2 trasformation matrix. mRowOffset — Gap (in user units) between individual cell references (for an array of cell references) along the X axis. mColOffset — Gap (in user units) between individual cell references (for an array of cell references) along the Y axis.
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sTPARAMS**
Objective:	Represents the transformation parameters (scale, rotation, reflection) specific to a cell reference or an array reference.
Members:	mMirror 'X': reflection along X axis. 'Y': refelection along Y axis. 'N': no reflection. mScale Scale transformation to be applied to a cell for a particular reference. mAngle Rotation transformation (in degrees) to be applied to a cell for a particular reference.
Methods:	none.
Notes:	1.
Last Revision:	0x5622522

Struct	**sQCKAREF**	
Objective:	Represents a single instance of an array of cell references in the CAD data space. (user units)	
Members:	mMatrix	true: Transformation parameters are represented as a 2x2 transformation matrix (mTMatrix). false: Transformation parameters are represented as scale, rotation and reflection (mTParams).
	mRefName	Name of the cell being referenced.
	mParentName	Name of the parent cell to which this reference belongs.
	mSize	Dimensions of the array, number of rows (1st element) and number of columns (2nd element) of individual cell references forming the array.
	mInsertion	X,Y co-ordinates of the point at which the array is inserted.
	mExtentsXY	Extents box (in user units) of the complete array reference.
	mTMatrix	Transformation parameters are represented as a 2x2 transformation matrix.
	mTParams	Transformation parameters are represented as scale, rotation and reflection.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKHAREF**	
Objective:	Represents a single instance of an array of cell references in the CAD data space just like sQCKAREF along with extents information about its parents and grandparents all the way to the TOP cell.	
Members:	mNumParents	Number of parents and grandparents referencing this instance of a cell reference all the way to the topmost cell.
	mParentTree	Array of extents information corrsponding to each parent or grandparent. (contains mNumParents elements).
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKTEXT**	
Objective:	Represents a single text item present in the CAD data space. (user units)	
Members:	mMirror	'X': Text is reflected along X axis during insertion. 'Y' : Text is reflected along Y axis. 'N': Text is not reflected.
	mJustify	Justification of the text during insertion. (eJustifyTextType)
	mFontID	An index into Qckvu3 database representing the font to be used while rendering the text.
	mLayer	Layer number of the layer on which the text is present.
	mTextType	Type number corresponding to this text item.
	mCellName	Name of the cell to which this text item belongs.
	mTextString	String of text characters to be displayed on Qckvu3 screen for this text item.
	mInsertion	X,Y co-ordinates of the point at which the text is inserted.
	mScale	Scale transformation applied to the text during insertion.
	mAngle	Rortation is degrees applied to the text during insertion.
	mExtentsXY	X,Y co-ordinates of extents box of the text item. (contains 8 doubles representing the 4 anchor points of the extents box.)
Methods:	none.	
Notes:	1.	
Last Revision:	0x5636D8F	

Struct	**sQCKVECTOR**	
Objective:	Represent a single generic vector data item (boundary, path, text, cell reference or an array reference) present in the CAD data space. (user units)	
Members:	mType	'B': this object represents a boundary (mBoundary) 'P': this object represents a path (mPath) 'S': this object represents a cell reference (mSref) 'A': this object represents an array reference (mAref) 'T': this object represents a text (mText) 's': this object represents a cell reference with hierarchy information (mHSref) 'a': this object represents an array reference with hierarchy information (mHAref)
	mNestingLevel	The nesting level relative to the view cell where this data item is positioned. (1: immediate child of the current view cell, -1: nesting level unavailable/disabled.)
	mBoundary	Data describing a boundary.
	mPath	Data describing a path.
	mSref	Data describing a cell reference.
	mAref	Data describing an array of cell references.
	mText	Data describing a text.
	mHAref	Data describing a cell reference with hierarchy information.
	mHSref	Data describing an array of cell references with hierarchy information.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKINFOVECTOR**	
Objective:	Contains information about a single vector data item (boundary, path, text, cell reference or an array reference) selected by the user via the Qckvu3 Info feature.	
Members:	mVertexID	Index (into the array of X,Y co-ordinates of the item represented by this object) indicating the specific vertex selected by the user via the Qckvu3 Info tool.
Methods:	none.	
Notes:	1.	
Last Revision:	0x5622522	

Struct	**sQCKQUERYSETTINGS**	
Objective:	Contains parameters to customize a query. (GetDataVector)	
Members:	mPathAsPath	true: Get 'Path' vector data as Paths. (default) false: Get 'Path' vector data as Boundaries.
	mGetText	true: Get 'Text' vector data. (default) false: Do not get 'Text' vector data.
	mNestingLevel	Return data only on this nesting level. (default nestingLevelALL)
	mWindow	Return data that crosses this window.
Methods:	Clear	Restore contents to default values.
Notes:	1.	
Last Revision:	0x56238E4	

Method	**sQCKQUERYSETTINGS :: Clear**
Objective:	Restore contents to default values.
Prototype:	`void Clear();`
Parameters:	none.
Return Values:	none.
Notes:	1.
Last Revision:	0x56238E4
Sample Code:	

REVISION HISTORY

Index > *Version History*

API Version	0x56238E4
Framework Version	1.1.0 (Mar 25, 2009)
Summary of Changes	
API Change	GetDataVector has a new parameter of type sQCKQUERYSETTINGS to enable the user to customize the query.
New Data Structure	sQCKQUERYSETTINGS contains parameters for cutomizing a query.
Document Changes	GetDataVector, sQCKQUERYSETTINGS

API Version	0x5636D8F
Framework Version	1.2.0 (Apr 02, 2009)
Summary of Changes	
API Change	sQCKTEXT supports a new member mExtentsXY to store extents box of the text item.
Document Changes	sQCKTEXT

API Version	0x5638C7F
Framework Version	1.3.0 (Apr 10, 2009)
Summary of Changes	
API Change	IQckExtractSrvc :: SelectPolygonArea is now IQckWindowSrvc :: SelectPolygonArea.
New Method	New method IQckEventNotify :: OnShowDialog (WINDOWS only)
New Method	New overload for method IQckDatabaseSrvc :: GetLayerList
New Method	New overload for method IQckDatabaseSrvc :: FreeLayerList
New Method	New Method IQckGraphicsSrvc::GetPixelSize
Document Changes	SelectPolygonArea, OnShowDialog, GetLayerList, FreeLayerList, GetPixelSize

API Version	0x5653E56
Framework Version	1.4.0 (May 21, 2009)
Summary of Changes	
API Change	Redraw has a new parameter zoomHome.
Bug Fix	SetViewCell does a 'zoom home' when refresh is true. Previously it did a Redraw which was causing the newly opened cell to mismatch with the window extents.
New Method	IQckSrvcMgr::GetNotificationHandler
Document Changes	Redraw, SetViewCell, GetNotificationHandler

API Version	0x56559B6
Framework Version	1.5.0 (May 28, 2009)
Summary of Changes	

API Change	sQCKSREF has two new parameters mMatrix and union of mTMatrix and mTParams.
Document Changes	sQCKSREF