NOKIA

Nokia 9000/9110 Development Environment Tool Developer's guide

Last Modified: June 10, 1999

Table of contents

Table of Figures

1 Introduction

This document will explain the details and expose the architecture of Nokia 9000 Developers Environment tools. It will also guide you through the creation process of a tool shell. In that tool shell, you will be able to insert functionality in order to help a Nokia 9000 GEOS developer perform certain GEOS programming tasks faster and easier.

2 Getting started

This guide explains what is, and how to create a tool shell. In that tool shell, you will include all extra code required for implementing the tool. A tool shell has the following responsibilities:

- To register itself in the system so that the N9KDE will be able to recognize it.
- To implement an interface that will allow the N9KDE to interact with it.
- To define a starting point in which the tool will be asked to launch execution.
- To instantiate an N9KDE object that will allow the tool to interact with the N9KDE.

Building a tool shell requires knowledge of ActiveX/COM technology. Every tool is in fact an ActiveX dll. An ActiveX dll is different from a conventional .dll because it is loaded through COM. The use of an ActiveX .dll allows the N9KDE to dynamically load any tool by means of a COM interface.

There are two ways of creating a tool. One method requires knowledge of Active Template Library (ATL) from Microsoft, as well as Visual C++ 6.0. Use of ATL wizards allows programmers to create COM enabled objects much more rapidly by hiding COM's complexity. Another method is building the tool from scratch, which requires more time and a deeper knowledge of COM programming.

2.1 Architecture of a tool

A tool is composed of a COM object, CComponent, supporting an interface, IComponent, that is known to the N9KDE. At first, the N9KDE "awakens" the tool by creating an instance of the CComponent COM object, by using its ProgID. The ProgID is a string that identifies the class and that resides in the system registry. With the ProgID, the N9KDE will be able to find the object's Class ID (CLSID) and create an instance of it. Once the object is instantiated, the N9KDE queries the object for its IComponent interface by its Interface ID (IID). The IComponent's IID is known to the N9KDE by the following GUID:

IComponent IID : {6A4EC0C3-AFA7-11D2-8F08-000000000000}

Each one and every tool must support the IComponent interface so to allow the N9KDE to launch and interact with it. The IComponent is identified with its IID (defined above).

This IID must be used in the Interface Definition Language (.IDL) file of the tool, which is accessed by the tool's COM.

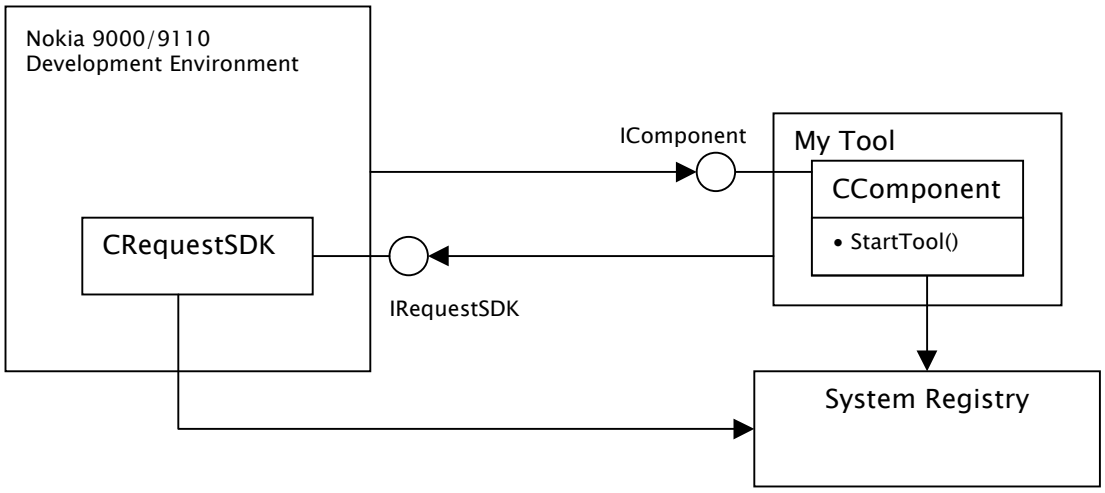


Figure 1 - Overview of the N9KDE-Tool architecture

2.2 N9KDE-Tool relationship

2.2.1 Tool registration

The tool must be registered in the user's system using the RegSrv32 utility. This allows the tool (which is an ActiveX .dll) to be recognized by the system as a COM object server. The registration process stores the CLSID, the ProgID as well as the IID within the system's registry. However, a tool needs to do a bit more to be visible to the N9KDE. It must also add entries to a specific key within the system registry. It must first create a key inside the "HKEY_CURRENT_USER\Software\Nokia\N9KDE\1.0\Toolbox" key. The key must have the tool's official name (for example: Flashing Note Toolbox) (Max. 50 chars). Inside that key, it must create string value titled "ProgID" in which it will include its Component's class ProgID.

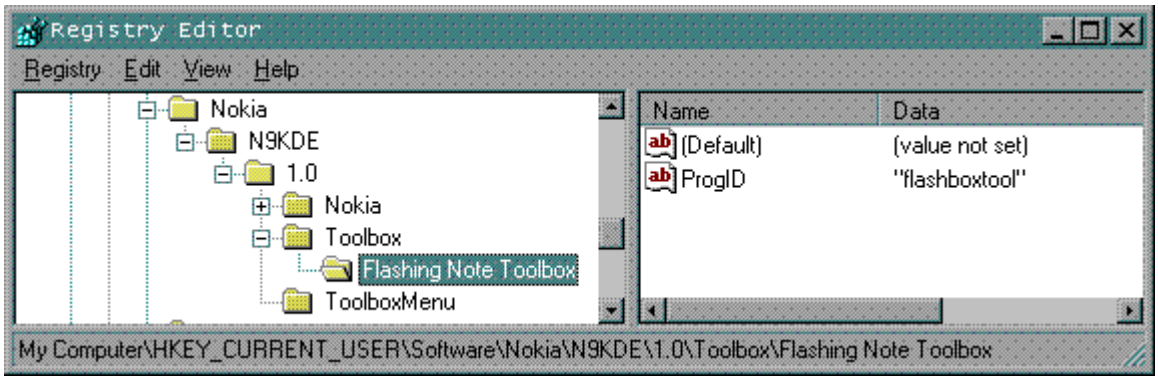


Figure 2 - Registry Editor view of the Toolbox key

2.2.2 Loading a tool from the N9KDE

Once the tool is properly registered, the N9KDE is able to reach for it by looking in the "HKEY_CURRENT_USER\Software\Nokia\N9KDE\1.0\Toolbox" key. Each tool entry will be picked up by the N9KDE and displayed to the user when the Tools Manager is launched from the N9KDE.

A tool is launched from within the N9KDE, either in the Tools Manager dialog (using the "Start Tool Now" button) or from the menu. Once a tool has been selected, the N9KDE uses the ProgID string that was initially read from the tool's registry entry, and with it, locates the tool's CLSID within the system registry. With the tool's CLSID, the N9KDE is then able to create an instance of the tool and query its IComponent interface.

Once the N9KDE holds a pointer to the IComponent interface, it automatically calls its only method, StartTool(), to launch the tool.

The tool has then the responsibility to create an instance of the CRequestSDK COM object in order to communicate with the N9KDE.

As soon as the tool terminates its execution, it must release the CRequestSDK object and dismiss itself. The N9KDE will automatically release its CComponent object instance and return to normal operation.

2.3 Builing a tool using Visual C++ 6.0 and ATL

2.3.1 Creating the tool shell

To create a tool shell using ATL in Visual C++ is quite simple. First you will need to create an ATL project by invoking the ATL COM wizard:

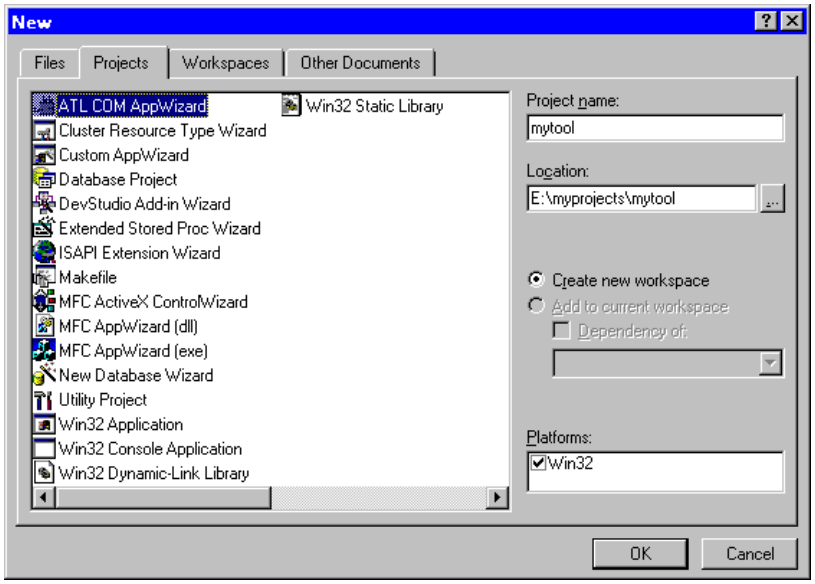
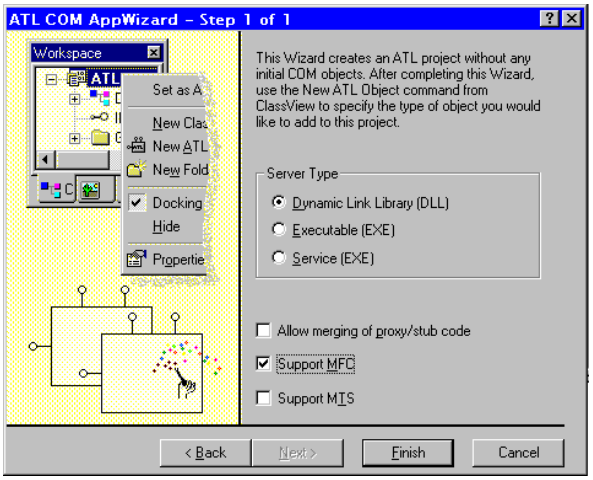


Figure 3 - Visual C++ 6.0 New project dialog view

Once launched, the wizard will display its configuration dialog box asking for the server type and other options. In our case, you will need to select Dynamic Link Library (DLL) option. Since tools have user interfaces, you might as well select the “Support MFC” option to ease creation of windows and other visual objects.

Figure 4 - ATL COM Wizard dialog



The next step is to create a COM object that will implement the point of contact for the N9KDE. This COM object will have to implement a specific interface that is known to the N9KDE. To create a simple COM object, reach for the "Insert" menu and select "New ATL Object". This will start the ATL Object Wizard:

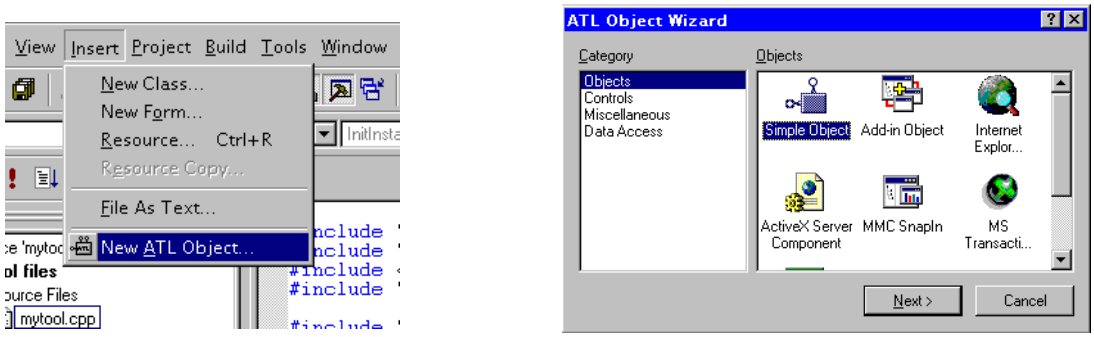


Figure 5 - Adding a new ATL object to the tool shell

Select the "Simple Object" object in the "Objects" category and click on "Next>". You'll be presented with a properties dialog. In the "Short Name" field, type Component. The other fields will fill up automatically. You may choose a different ProgID if you wish (no spaces). Then click on the "Attributes" tab, and set the properties as shown in the figure (**Threading Model**: Single; **Interface**: Custom; **Aggregation**: No)

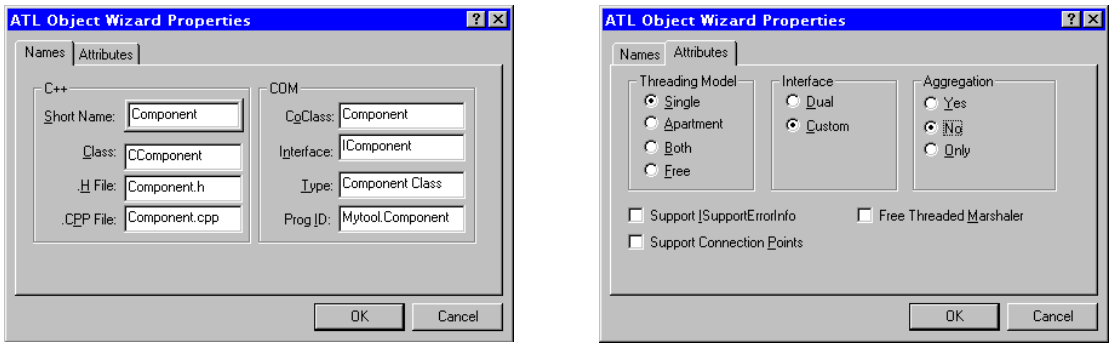


Figure 6 - Creating the CComponent COM object

Once you click on "Ok", the wizard will automatically add the Component.cpp and Component.h files to your project, as well as updating the tool's IDL file by adding references to the Component COM object.

From this point, you'll need to change the automatically created .IDL file, as well as adding a method (StartTool() method) to CComponent class.

To add a method to the CComponent class definition of your tool, click on the "Class View" tab of your workspace window, right-click on the IComponent "spoon"-like icon and choose "Add Method" option out of the floating menu. Create a method called StartTool with no parameters (refer to the figure) and click

Ok. This will automatically update your IComponent class definition as well as CComponent class definition.

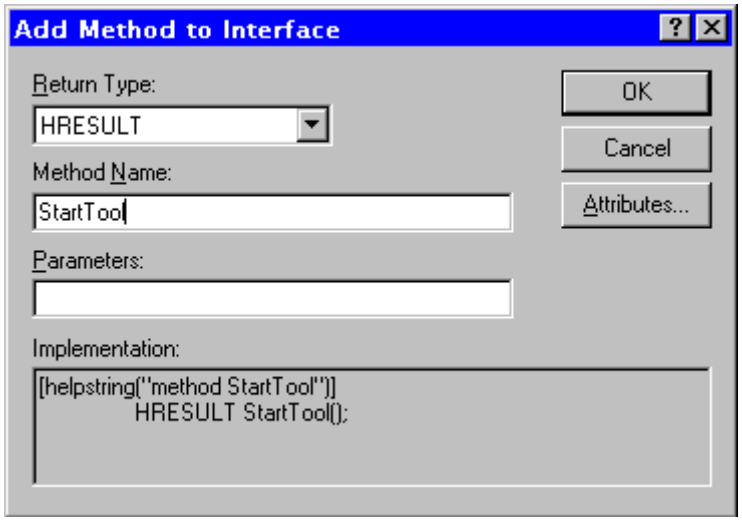


Figure 7 - Adding the StartTool() method the IComponent interface

Another important step is to change the UUID of the IComponent interface that has been automatically assigned by the ATL Object Wizard. The UUID has to be replaced by {6A4EC0C3-AFA7-11D2-8F08-000000000000} (instead of the {68532DB0-CF26-11D2-8F34-000000000000} UUID shown in this "My tool" example). This is required to allow the N9KDE to recognize your IComponent's interface.

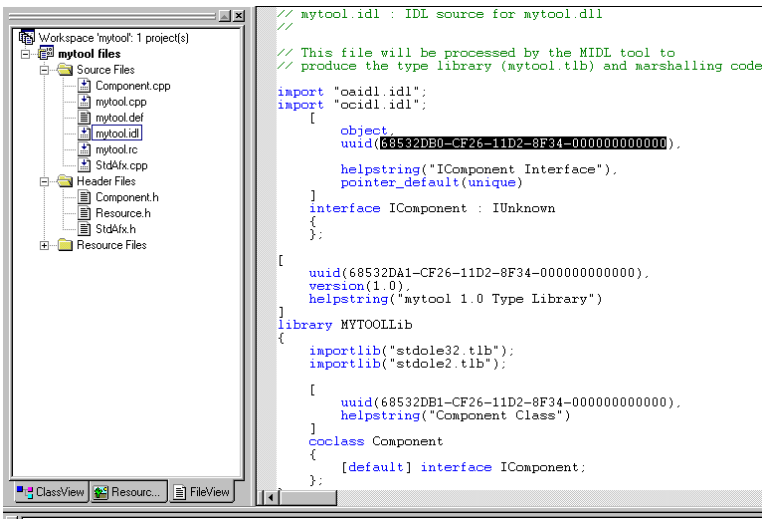


Figure 8 - Replacing the IComponent`s IID

2.3.2 Performing tool registration

As the last step for the tool shell creation, you'll need to add the registry entries that will make your tool visible to the N9KDE. All the registration process should be done once, usually when your tool will be installed for the first time on the user's system. To allow your tool to register within the N9KDE's registry keys, locate the tool's *STDAPI DllRegisterServer(void)* function (or your CwinApp application class' InitInstance() function) and add the following code:

```
HKEY   hk;
DWORD  dwDisp = 50; // Maximum tool's name size
unsigned char szProgID[] = "Sample"; // This is the tool's ProgID
unsigned char szToolName[] = "Sample Tool"; // This is the tool's readable name. This name will appear in the N9KDE
CString szRegDir;
DWORD  dwValueSize;

// Format the Registry Directory string
szRegDir.Format("Software\\Nokia\\N9KDE\\1.0\\Toolbox\\%s",szToolName);

// Add tool's key in the HKEY_CURRENT_USER\Software\Nokia\ N9KDE \1.0\Toolbox key
 RegCreateKeyEx( HKEY_CURRENT_USER,
                 (LPCTSTR)szRegDir,
                 0,
                 NULL,
                 REG_OPTION_NON_VOLATILE,
                 KEY_ALL_ACCESS,
                 NULL,
                 &hk,
                 &dwDisp
                 );

dwValueSize = strlen((char*)szProgID);
dwValueSize++; //Add NULL terminating character

// Add the ProgID sting value
RegSetValueEx( hk,
               "ProgID",
               0,
               REG_SZ,
               szProgID,
               dwValueSize
);
```

This portion of code will add the appropriate key entries in the N9KDE's tool registry section in order to allow the tool to be located by the N9KDE.

To register your tool, you need only invoke the RegSvr32 utility (from a command prompt):
>regsvr32 mytool.dll

Similarly, in the tool's *STDAPI DllUnregisterServer(void)* function (or your CWinApp application class' ExitInstance() function), you should add code to unregister your tool. Unregistring will remove the registry entries initially created when the tool first registered. Add the following code to remove the appropriate keys:

```
HKEY   hk;
CString sz;
CString szToolName = "Sample Tool";// This is the tool's readable name.

 // First Delete contents of Registry under ToolboxMenu Key
sz = "Software\\Nokia\\N9KDE\\1.0\\ToolboxMenu";

 RegOpenKeyEx(  HKEY_CURRENT_USER,
                (LPCTSTR)sz,
                0,
                KEY_ALL_ACCESS,
                &hk
             );

 RegDeleteKey(  hk,
              szToolName
             );

// Second Delete contents of Registry under Toolbox Key
sz = "Software\\Nokia\\N9KDE\\1.0\\Toolbox";

 RegOpenKeyEx(  HKEY_CURRENT_USER,
                (LPCTSTR)sz,
                0,
                KEY_ALL_ACCESS,
                &hk
             );
RegDeleteKey(  hk,
             szToolName
);
```

Figure 10 - Removing tool registry data

2.3.3 Integrating your tool code

Now that you have a tool shell, you must fill it with your own tool implementation code. In the StartTool method to your CComponent class definition, you must add your code in order to launch (or create) your own tool code.

```
// Component.cpp : Implementation of CComponent
#include "stdafx.h"
#include "Mytool.h"
#include "Component.h"

/////////////////////////////////////////////////////////////////////////////
// CComponent

STDMETHODIMP CComponent::StartTool()
{
        AFX_MANAGE_STATE(AfxGetStaticModuleState())
        // TODO: Add your implementation code here
        // Create an instance of your tool's main class here
        return S_OK;
}
```

2.4 Interacting with N9KDE

To interact with the N9KDE during tool execution, you must access the N9KDE's own IRequestSDK interface. In order for your tool's code to access N9KDE's IRequestSDK methods, you need to create an instance of the N9KDE's CRequestSDK object. Provided in this package are the files Nokia_i.h and Nokia_i.c that define the IRequestSDK interface. Note that it is important to include the Nokia_i.c and the Nokia_i.h files in the main .CPP file of your tool project. The Nokia_i.c file contains all UUIDs used in the N9KDE.

To create an instance of the N9KDE's, calls must be made to COM library functions with the appropriate parameters. First, it is required to call *CoGetClassObject* with the N9KDE's own class ID (CLSID_RequestSDK) in order to get its class factory. Then an instance of the RequestSDK class must be created by calling the class factory's *CreateInstance* function while trying to attach to that class's IUnknown interface. Then once a pointer to the class's IUnknown is obtained, the QueryInterface method from the latter must be invoked with the IRequestSDK's interface ID (IID_IRequestSDK) as a parameter, in order to finally reach for the N9KDE's interface (see sample code in figure 12).

In the code sample, the pCI pointer implements a pointer to the IRequestSDK interface, and provides access to all of the N9KDE's methods (refer to chapter 3).

```
#define _WIN32_DCOM
#include <objbase.h>
#include <initguid.h>
#include "Nokia_i.h"


…

        HRESULT hRes;
        IRequestSDK *pCI;
        IUnknown * pM;
        IClassFactory * pCF;


        // Create an instance of the CRequestSDK object
        hRes = CoGetClassObject(CLSID_RequestSDK, CLSCTX_ALL, NULL, IID_IClassFactory, reinterpret_cast<void **>(&pCF));

        if (SUCCEEDED(hRes))
        {
                // Invoke the class factory's Create Instance method to get a pointer to the object's IUnknown interface
                hRes = pCF->CreateInstance(NULL, IID_IUnknown, reinterpret_cast<void **>(&pM) );
                pCF->Release();
                if (pM != NULL)
                {
                        // Invoke the IUnknown 's Query Interface method to get a pointer to the IRequestSDK interface
                        hRes = pM->QueryInterface( IID_IRequestSDK, reinterpret_cast<void **>(&pCI) );
                        pM->Release();
                        if(pCI != NULL)
                        {
                                // pCI is now a valid pointer to the IRequestSDK interface
                        }
                        else
                        {
                                AfxMessageBox("TOOL: Unable to query SDK interface !\n");
                                return FALSE;
                        }
                }
                else
                {
                        AfxMessageBox("TOOL: Unable to create object !\n");
                        return FALSE;
                }
        }
        else
        {
                AfxMessageBox("TOOL: Unable to get a pointer to the SDK interface !\n");
                return FALSE;
        }

        // Release SDK's CRequestSDK COM object
        pCI->Release();

…
```

3 Programmers Reference

The N9KDE's IRequestSDK interface provides a full set of methods that can be used by the tool to read or write project related data attributes, update source code blocks. All code examples that are provided in this section refers to the code sample described in figure 12.

Method	Description
get_Heap(long *pVal)	Obtains the heap size (in K) from the N9KDE.
put_Heap(long newVal)	Sets the current value of the heap in the tool.
get_TokenID(long *pVal)	Gets the token ID from the N9KDE.
put_TokenID(long newVal)	Sets the token ID in the N9KDE for the project.
get_ProjectName(BSTR *newVal)	Gets the project name
get_ProjectDir(BSTR *newVal)	Gets the project directory
get_TypeGP(BSTR *pVal)	Tells the tool what is included in the "TYPE" line.
put_TypeGP(BSTR *newVal)	Sets the "TYPE" line from the project's source code.
get_Libraries(BSTR *pVal)	Tells the tool what are the current libraries used in the project
put_Libraries(BSTR *newVal)	Sets the libraries for the project
get_ResourcesGP(BSTR *pVal)	Tells the tool what the resource list is.
put_ResourcesGP(BSTR *newVal)	Sets the resource list for the project
get_ExportGP(BSTR *pVal)	Tells the tool what the export list is.
put_ExportGP(BSTR *newVal)	Sets the export list for the project
get_TokenChars(BSTR *pVal)	Gets the 4-character token char from the project.
put_TokenChars(BSTR *newVal)	Gets the 4-character token char for the project.
get_ConstantsGOH(BSTR *pVal)	Gets the list of constants from the project
put_ConstantsGOH(BSTR *newVal)	Sets the list of constants for the project
get_DataTypesGOH(BSTR *pVal)	Gets the data type block from the project.
put_DataTypesGOH(BSTR *newVal)	Sets the project's data type block.
get_ClassDefnGOH(BSTR *pVal)	Gets the project's class definition list block.
put_ClassDefnGOH(BSTR *newVal)	Sets the project's class definition list block.
get_IncludeGOC(BSTR *pVal)	Gets the project's list of include files (within the block).
put_IncludeGOC(BSTR *newVal)	Sets the project's list of include files (within the block).
get_ClassDeclGOC(BSTR *pVal)	Gets the class declarations within the project.
put_ClassDeclGOC(BSTR *newVal)	Sets the class declarations within the project
get_GlobalsGOC(BSTR *pVal)	Gets the project's global variable list.
put_GlobalsGOC(BSTR *newVal)	Sets the project's global variable list.
get_DisplayNameGOC(BSTR *pVal)	Gets the project's display name (the display name is found under the application main Icon, in left status pane).
put_DisplayNameGOC(BSTR *newVal)	Sets the project's display name (the display name is found under the application main Icon, in left status pane).
get_ResourcesGOC(BSTR *newVal)	Gets the project's resource list from the main GOC file.
get_ObjectsGOC(BSTR *newVal)	Gets the project's objects list from the main GOC file
get_MethodsGOC(BSTR *newVal)	Gets the project's methods list from the main GOC file
get_Version(float *pVal)	Gets the N9KDE's version number.
get_MainWinHandle(short *pVal)	Gets the N9KDE's main window handle.
Get_Block(BSTR *szStart, BSTR *szEnd, BSTR *szContents);	Gets a block of source code found between a set of delimiters strings (start,end) among the main GOC, GOH and GP files.
Set_Block(BSTR *szStart, BSTR *szEnd, BSTR *szContents);	Sets a block of source code found between a set of delimiters strings (start,end) among the main GOC, GOH and GP files.

3.1 get_Heap

get_Heap(long *pVal)

The Get Heap function reaches for the Heap size (in KB) information that is residing within the current project's .GP file. It can be used, for instance, to recalculate a newer value considering the new resources added by the tool, to the project.
Example:

```
Long m_EditHeap;
…
pCI->get_Heap(&m_EditHeap); // pCI is a pointer to the IRequestSDK interface
```

3.2 put_Heap

put_Heap(long newVal)

The put_Heap function writes a new Heap size in (KB) information to the current project's .GP file. It can be used, for instance, once a new value of Heap size has been recalculated and needs to be included in the project.

Example:

```
Long m_EditHeap;
…

// pCI is a pointer to the IRequestSDK interface
pCI->get_Heap(&m_EditHeap); // pCI is a pointer to the IRequestSDK interface

m_EditHeap += 200; // add 200 KB to the original Heap value

// Overwrite newer Heap size value to the current project:
pCI->put_Heap(m_EditHeap);
```

3.3 get_TokenID

get_TokenID(long *pVal)

The get_TokenID function reads for the project’s TokenID (tokenid) that resides in the .GP file. The TokenID identifies a unique token in GeoManager's token database file. It must be a number corresponding to the programmer's manufacturer ID number.

```
Long m_TokenID;
…

// pCI is a pointer to the IRequestSDK interface
pCI->get_TokenID(&m_TokenID);
```

3.4 put_TokenID

put_TokenID(long newVal)

The put_TokenID function overwrites the project’s TokenID (tokenid) that resides in the .GP file. The TokenID identifies a unique token in GeoManager's token database file. It must be a number corresponding to the programmer's manufacturer ID number.

```
Long m_TokenID;

…

m_TokenID = 2222;

// pCI is a pointer to the IRequestSDK interface
pCI->put_TokenID(m_TokenID);
```

3.5 get_ProjectName

get_ProjectName(BSTR *newVal)

The get_ProjectName function reads the name of the currently opened project.

```
CString szPrjName; // CString is a MFC specific class
BSTR  bszTemp;

…
// pCI is a pointer to the IRequestSDK interface
pCI->get_ProjectName(&bszTemp);

// Get the project name into a CString object
fromBSTR(bszTemp,szPrjName);
}
…
void fromBSTR(BSTR bszStr, CString & szValue)
{
                szValue = bszStr;
}
```

3.6 get_ProjectDir

get_ProjectDir(BSTR *newVal)

The get_ProjectDir function reads the directory name of the currently opened project.

```
CString szDirName; // CString is a MFC specific class
BSTR  bszTemp;

…
// pCI is a pointer to the IRequestSDK interface
pCI->get_ProjectDir(&bszTemp);

// Get the project name into a CString object
fromBSTR(bszTemp,szDirName);
}
…
void fromBSTR(BSTR bszStr, CString & szValue)
{
                szValue = bszStr;
}
```

3.7 get_TypeGP

get_TypeGP(BSTR *pVal)

This function reads the *type* field in the .GP file (example : type appl, process, single). The *type* field in the parameters file designates certain characteristics of the geode being compiled. These attributes correspond to the GeodeAttrs type and determine how the Glue linker will put the geode together.

```
CString szTypeGP; // CString is a MFC specific class
BSTR  bszTemp;

…

// pCI is a pointer to the IRequestSDK interface
pCI->get_TypeGP(&bszTemp);

// Get the project's type field into a CString object
fromBSTR(bszTemp, szTypeGP);
}
…
void fromBSTR(BSTR bszStr, CString & szValue)
{
        szValue = bszStr;
}
```

3.8 put_TypeGP

put_TypeGP(BSTR *newVal)

This function overwrites the *type* field in the .GP file (example : type appl, process, single). The *type* field in the parameters file designates certain characteristics of the geode being compiled. These attributes correspond to the GeodeAttrs type and determine how the Glue linker will put the geode together.

```
CString szTypeGP; // CString is a MFC specific class
BSTR  bszTemp;

…

// pCI is a pointer to the IRequestSDK interface
pCI->get_TypeGP(&bszTemp);

// Get the project's type field into a CString object
fromBSTR(bszTemp, szTypeGP);

// Add "single" attribute to the type field, if it isn't already there
if ( szTypeGP.Find("single") == -1)
        szTypeGP += ", single";

bszTemp = toBSTR(szTypeGP);

pCI->put_TypeGP(&bszTemp);


…

void fromBSTR(BSTR bszStr, CString & szValue)
{
        szValue = bszStr;
}
BSTR toBSTR(CString szValue)
{
        return szValue.AllocSysString();
}
```

3.9 get_Libraries

get_Libraries(BSTR *pVal)

This function reads the *library* fields in the .GP file. This field specifies the libraries that are used by the project. The returned value points to an array of strings.

```
BSTR bszTemp;
CStringArray m_csaLibraries;
Cstring szLibraries;

…

// Initialize the string array
m_csaLibraries.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_Libraries(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaLibraries);

// Get all strings into one single string: szLibraries
szLibraries = fromArraytoStr(m_csaLibraries);
}

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.10 put_Libraries

put_Libraries(BSTR *newVal)

This function overwrites the *library* section delimited by the #//{{LIBRARIES-GP keywords in the .GP file. This field specifies the libraries that are used by the project. The returned value points to an array of strings.

The following example demonstrates the use of the put_Libraries method.

```
BSTR bszTemp;
CStringArray m_csaLibraries;
BOOL bFound = FALSE;

…

// Initialize the string array
m_csaLibraries.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_Libraries(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaLibraries);


// Add “library ui” string to the library section, if it isn’t already there
for( int i = 0; i < m_csaLibraries.GetSize(); i++ )
{
      if (m_csaLibraries.GetAt(i).Find(“library ui”) != -1)
      {
          bFound = TRUE;
          break;
      }
}
if (!bFound) m_csaLibraries.Add(“library ui\n”)

// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaLibraries);

// Write the updated library list
pCI->put_Libraries(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
                CString szTmp = bszStr;
                CString szRight, szLeft;
                int nLoc = -1;
                int i = 0;
                while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
                {
                                if( nLoc )        szLeft = szTmp.Left( nLoc );
                                else szLeft = "";

                                csa.Add( szLeft );
                                szTmp = szTmp.Mid( nLoc + 1);
                                i++;
                }
                csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
                CString szTmp;
                for( int i = 0; i < csa.GetSize(); i++ )
                {
                                if( i )szTmp += "\n";
                                szTmp += csa.GetAt(i);
                }
                return szTmp.AllocSysString();
}
```

3.11 get_ResourcesGP

get_ResourcesGP(BSTR *pVal)

This function reads the *resource* list delimited by the #//{{RESOURCES-GP keywords in the .GP file This field indicates to the Glue linker which named resources the project uses. The returned value points to an array of strings.

The following example demonstrates the use of the get_ResourcesGP method.

```
BSTR bszTemp;
CStringArray m_csaResourcesGP;
CString szResourcesGP;

…

// Initialize the string array
m_csaResourcesGP.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ResourcesGP (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaResourcesGP);

// Get all strings into one single string: szResourcesGP
szResourcesGP = fromArraytoStr(m_csaResourcesGP);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.12 put_ResourcesGP

put_ResourcesGP(BSTR *newVal)

This function overwrites the *resources* list delimited by the #//{{RESOURCES-GP keywords in the .GP file This field indicates to the Glue linker which named resources the project uses. The returned value points to an array of strings.

The following example demonstrates the use of the put_ResourcesGP method.

```
BSTR bszTemp;
CStringArray m_csaResourcesGP;
BOOL bFound = FALSE;

…

// Initialize the string array
m_csaResourcesGP.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_ResourcesGP(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaResourcesGP);


// Add “resource INTERFACE ui-object” string to the resources list, if it isn’t already there
for( int i = 0; i < m_csaResourcesGP.GetSize(); i++ )
{
      if (m_csaResourcesGP.GetAt(i).Find(“resource INTERFACE ui-object”) != -1)
      {
          bFound = TRUE;
          break;
      }
}
if (!bFound) m_csaResourcesGP.Add(“resource INTERFACE ui-object\n”)

// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaResourcesGP);

// Write the updated Resources list
pCI->put_ResourcesGP(&bszTemp);
…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
                CString szTmp = bszStr;
                CString szRight, szLeft;
                int nLoc = -1;
                int i = 0;
                while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
                {
                                if( nLoc )        szLeft = szTmp.Left( nLoc );
                                else szLeft = "";

                                csa.Add( szLeft );
                                szTmp = szTmp.Mid( nLoc + 1);
                                i++;
                }
                csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
                CString szTmp;
                for( int i = 0; i < csa.GetSize(); i++ )
                {
                                if( i )szTmp += "\n";
                                szTmp += csa.GetAt(i);
                }
                return szTmp.AllocSysString();
}
```

3.13 get_ExportGP

get_ExportGP(BSTR *pVal)

This function reads the exported class list delimited by the #//{{EXPORT-GP keywords in the .GP file. This field identifies routines usable by geodes other than the one being compiled. This field is also used to export classes defined in a .goc or .goh file. The returned value points to an array of strings.

The following example demonstrates the use of the get_ExportGP method.

```
BSTR bszTemp;
CStringArray m_csaExportGP;
CString szExportGP;

…

// Initialize the string array
m_csaExportGP.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ExportGP (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaExportGP);

// Get all strings into one single string: szExportGP
szExportGP = fromArraytoStr(m_csaExportGP);

…

CString fromArraytoStr(CStringArray & csa)
{
		// concatenate the strings, delimited by "\n"
		CString szTmp;
		for( int i = 0; i < csa.GetSize(); i++ )
		{
			szTmp += csa.GetAt(i);
			szTmp += "\n";
		}
		return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
		CString szTmp = bszStr;
		CString szRight, szLeft;
		int nLoc = -1;
		int i = 0;
		while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
		{
			if( nLoc )	szLeft = szTmp.Left( nLoc );
			else szLeft = "";

			csa.Add( szLeft );
			szTmp = szTmp.Mid( nLoc + 1);
			i++;
		}
		csa.Add( szTmp );
}
```

3.14 put_ExportGP

put_ExportGP(BSTR *newVal)

This function overwrites the export list section delimited by the #//{{EXPORT-GP keywords in the .GP file. This field identifies routines usable by geodes other than the one being compiled. This field is also used to export classes defined in a .goc or .goh file. The returned value points to an array of strings.

The following example demonstrates the use of the put_ExportGP method.

```
BSTR bszTemp;
CStringArray m_csaExportGP;

…

// Initialize the string array
m_csaExportGP.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_ExportGP(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaExportGP);


// Add “export newVisContentClass” string to the exported class list
m_csaExportGP.Add(“export newVisContentClass \n”)

// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaExportGP);

// Write the updated export list
pCI->put_ExportGP(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
		CString szTmp = bszStr;
		CString szRight, szLeft;
		int nLoc = -1;
		int i = 0;
		while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
		{
			if( nLoc )	szLeft = szTmp.Left( nLoc );
			else szLeft = "";

			csa.Add( szLeft );
			szTmp = szTmp.Mid( nLoc + 1);
			i++;
		}
		csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
		CString szTmp;
		for( int i = 0; i < csa.GetSize(); i++ )
		{
			if( i )szTmp += "\n";
			szTmp += csa.GetAt(i);
		}
		return szTmp.AllocSysString();
}
```

3.15 get_TokenChars

get_TokenChars(BSTR *pVal)

This function reads the *tokenchar* field. That field identifies a unique token in GeoManager's token database file. The tokenchars field is a string of four characters that identifies the geode's token.

The following example demonstrates the use of the get_TokenChars method.

```
CString szTokenChars; // CString is a MFC specific class
BSTR  bszTemp;

…
// pCI is a pointer to the IRequestSDK interface
pCI->get_TokenChars(&bszTemp);

// Get the project name into a CString object
fromBSTR(bszTemp, szTokenChars);
}
…
void fromBSTR(BSTR bszStr, CString & szValue)
{
                szValue = bszStr;
}
```

3.16 put_TokenChars

put_TokenChars(BSTR *newVal)

This function overwrites the *tokenchar* field. That field identifies a unique token in GeoManager's token database file. The tokenchars field must be a string of four characters that identifies the geode's token.

The following example demonstrates the use of the put_TokenChars method.

```
CString szTokenChars; // CString is a MFC specific class
BSTR  bszTemp;

…
szTokenChars = “MAPP”; // Let’s change the user’s geode token to “MAPP” – must be four chars.

bszTemp = toBSTR(szTokenChars);

// pCI is a pointer to the IRequestSDK interface
pCI->put_TokenChars(&bszTemp);


…

BSTR MainToolDialog::toBSTR(CString szValue)
{
        return szValue.AllocSysString();

}
```

3.17 get_ConstantsGOH

get_ConstantsGOH(BSTR *pVal)

This function reads the geode's constants list delimited by the //{{CONSTANTS-GOH keywords in the .GOH file. This field identifies constants defined for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_ConstantsGOH method.

```
BSTR bszTemp;
CStringArray m_csaConstantsGOH;
CString szConstantsGOH;

…

// Initialize the string array
m_csaConstantsGOH.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ConstantsGOH (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaConstantsGOH);

// Get all strings into one single string szConstantsGOH:
szConstantsGOH = fromArraytoStr(m_csaConstantsGOH;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.18 put_ConstantsGOH

put_ConstantsGOH(BSTR *newVal)

This function overwrites the geode's constants list delimited by the //{{CONSTANTS-GOH keywords in the .GOH file. This field identifies constants defined for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the put_ConstantsGOH method.

```
BSTR bszTemp;
CStringArray m_csaConstantsGOH;

…

// Initialize the string array
m_csaConstantsGOH.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_ConstantsGOH(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaConstantsGOH);

////// Add constants ////////////////////////////////////////////////////////////
// Add "#define MIN_ITERATION   4"  constant
m_csaConstantsGOH.Add("#define MIN_ITERATION   4\n")
// Add constant (cMaxIteration) in constant list by adding string "const short cManIteration = 20;"
m_csaConstantsGOH.Add("const short cMaxIteration = 20;\n")


// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaConstantsGOH);

// Write the updated constant list
pCI->put_ConstantsGOH(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )szTmp += "\n";
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();
}
```

3.19 get_DataTypesGOH

get_DataTypesGOH(BSTR *pVal)

This function reads the geode's data types list delimited by the //{{DATATYPES-GOH keywords in the .GOH file. This field identifies data types defined for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_DataTypesGOH method.

```
BSTR bszTemp;
CStringArray m_csaDataTypesGOH;
CString szDataTypesGOH;

…

// Initialize the string array
m_csaDataTypesGOH.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_DataTypesGOH (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaDataTypesGOH);

// Get all strings into one single string szDataTypesGOH:
szDataTypesGOH = fromArraytoStr(m_csaDataTypesGOH;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.20 put_DataTypesGOH

put_DataTypesGOH(BSTR *newVal)

This function overwrites the geode's data types list delimited by the //{{DATATYPES-GOH keywords in the .GOH file. This field identifies data types defined for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the put_DataTypesGOH method.

```
BSTR bszTemp;
CStringArray m_csaDataTypesGOH;

…

// Initialize the string array
m_csaDataTypesGOH.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_DataTypesGOH(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaDataTypesGOH);


////// Add a data type ////////////////////////////////////////////////////////////
// Add AppError enum to the DataTypes section
m_csaDataTypesGOH.Add("typedef enum {\n");
m_csaDataTypesGOH.Add("       APP_NO_ERROR, \n");
m_csaDataTypesGOH.Add("       APP_WARNING\n");
m_csaDataTypesGOH.Add("       APP_SEVERE_ERROR\n");
m_csaDataTypesGOH.Add("} ApiError; \n");


// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaDataTypesGOH);

// Write the updated data types list
pCI->put_DataTypesGOH(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )szTmp += "\n";
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();
}
```

3.21 get_ClassDefnGOH

get_ClassDefnGOH(BSTR *pVal)

This function reads the geode's class definitions list delimited by the //{{CLASSDEFN-GOH keywords in the .GOH file. This field identifies various class definitions defined for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_ClassDefnGOH method.

```
BSTR bszTemp;
CStringArray m_csaClassDefnGOH;
CString szClassDefnGOH;

…

// Initialize the string array
m_csaClassDefnGOH.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ClassDefnGOH (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaClassDefnGOH);

// Get all strings into one single string szClassDefnGOH:
szClassDefnGOH = fromArraytoStr(m_csaClassDefnGOH;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.22 put_ClassDefnGOH

put_ClassDefnGOH(BSTR *newVal)

This function overwrites the geode's class definitions list delimited by the //{{CLASSDEFN-GOH keywords in the .GOH file. This field identifies various class definitions defined for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the put_ClassDefnGOH method.

```
BSTR bszTemp;
CStringArray m_csaClassDefnGOH;

…

// Initialize the string array
m_csaClassDefnGOH.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_ClassDefnGOH(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaClassDefnGOH);


////// Add a class definition  ////////////////////////////////////////////////////////
// Add VisClass definition "class NewVisContentClass"  to the class definition section
m_csaClassDefnGOH.Add("@class NewVisContentClass, VisContentClass;\n");
m_csaClassDefnGOH.Add("@endc;\n");


// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaClassDefnGOH);

// Write the updated class definition list
pCI->put_ClassDefnGOH(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )szTmp += "\n";
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();
}
```

3.23 get_IncludeGOC

get_IncludeGOC(BSTR *pVal)

This function reads the geode's #include section delimited by the //{{INCLUDE-GOC keywords in the .GOC file. This field identifies #include definitions for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_IncludeGOC method.

```
BSTR bszTemp;
CStringArray m_csaIncludeGOC;
CString szIncludeGOC;

…

// Initialize the string array
m_csaIncludeGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_IncludeGOC (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaIncludeGOC);

// Get all strings into one single string szIncludeGOC:
szIncludeGOC = fromArraytoStr(m_csaIncludeGOC;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.24 put_IncludeGOC

put_IncludeGOC(BSTR *newVal)

This function overwrites the geode's #include section delimited by the //{{INCLUDE-GOC keywords in the .GOC file. This field identifies #include definitions for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the put_IncludeGOC method.

```
BSTR bszTemp;
CStringArray m_csaIncludeGOC;

…

// Initialize the string array
m_csaIncludeGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_IncludeGOC(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaIncludeGOC);


////// Add a class definition  ////////////////////////////////////////////////////////
// attach newly created file "example.goh" to the project
m_csaIncludeGOC.Add("@include <example.goh>\n");

// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaIncludeGOC);

// Write the updated #include definition list
pCI->put_IncludeGOC(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
	CString szTmp = bszStr;
	CString szRight, szLeft;
	int nLoc = -1;
	int i = 0;
	while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
	{
		if( nLoc )	szLeft = szTmp.Left( nLoc );
		else szLeft = "";

		csa.Add( szLeft );
		szTmp = szTmp.Mid( nLoc + 1);
		i++;
	}
	csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
	CString szTmp;
	for( int i = 0; i < csa.GetSize(); i++ )
	{
		if( i )szTmp += "\n";
		szTmp += csa.GetAt(i);
	}
	return szTmp.AllocSysString();
}
```

3.25 get_ClassDeclGOC

get_ClassDeclGOC(BSTR *pVal)

This function reads the geode's class declaration section delimited by the //{{CLASSDECL-GOC keywords in the .GOC file. This field identifies the various class declarations for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_ClassDeclGOC method.

```
BSTR bszTemp;
CStringArray m_csaClassDeclGOC;
CString szClassDeclGOC;

…

// Initialize the string array
m_csaClassDeclGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ClassDeclGOC (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaClassDeclGOC);

// Get all strings into one single string szClassDeclGOC:
szClassDeclGOC = fromArraytoStr(m_csaClassDeclGOC;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.26 put_ClassDeclGOC

put_ClassDeclGOC(BSTR *newVal)

This function overwrites the geode's class declaration section delimited by the //{{CLASSDECL-GOC keywords in the .GOC file. This field identifies the various class declarations for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the put_ClassDeclGOC method.

```
BSTR bszTemp;
CStringArray m_csaClassDeclGOC;

…

// Initialize the string array
m_csaClassDeclGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_ClassDeclGOC(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaClassDeclGOC);


////// Add a class declaration  ////////////////////////////////////////////////////////
// add class declaration for NewVisContentClass
m_csaClassDeclGOC.Add("@classdecl NewVisContentClass; \n");

// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaClassDeclGOC);

// Write the updated class declaration list
pCI->put_ClassDeclGOC(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )szTmp += "\n";
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();
}
```

3.27 get_GlobalsGOC

get_GlobalsGOC(BSTR *pVal)

This function reads the geode's global variables section delimited by the //{{GLOBALS-GOC keywords in the .GOC file. This field identifies various global variables definitions for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_GlobalsGOC method.

```
BSTR bszTemp;
CStringArray m_csaGlobalsGOC;
CString szGlobalsGOC;

…

// Initialize the string array
m_csaGlobalsGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_GlobalsGOC (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaGlobalsGOC);

// Get all strings into one single string szGlobalsGOC:
szGlobalsGOC = fromArraytoStr(m_csaGlobalsGOC;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.28 put_GlobalsGOC

put_GlobalsGOC(BSTR *newVal)

This function overwrtites the geode's global variables section delimited by the //{{GLOBALS-GOC keywords in the .GOC file. This field identifies various global variable definitions for the geode. The returned value points to an array of strings.

The following example demonstrates the use of the put_GlobalsGOC method.

```
BSTR bszTemp;
CStringArray m_csaGlobalsGOC;

…

// Initialize the string array
m_csaGlobalsGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI->get_GlobalsGOC(&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaGlobalsGOC);


////// Add a global variable  ////////////////////////////////////////////////////////
// add global variable bFlag to the project
m_csaGlobalsGOC.Add("BOOL bFlag = FALSE; \n");

// Transform the CStringArray into a BSTR
bszTemp = toBSTR(m_csaGlobalsGOC);

// Write the updated global variable definition list
pCI->put_GlobalsGOC(&bszTemp);

…

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}

BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )szTmp += "\n";
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();
}
```

3.29 get_DisplayNameGOC

get_DisplayNameGOC(BSTR *pVal)

This function reads the geodes's display name text moniker used within the FoamSubApplicationClass instantiation. The display name text moniker is found with the //{{DISPLAYNAME-GOC keyword, in the .GOC file. The display name is the string found under the geode's main status pane icon (situated at the leftmost part of the screen). The text moniker, which contains the string, is defined elsewhere in the .GOC file (refer to the Set_Block code example to know how to changes that string from the tool – or refer to the /* {{ICON LABEL BLOCK}} -- DO NOT ERASE */ keyword in the .GOC file).

The following example demonstrates the use of the get_DisplayNameGOC method.

```
CString szDisplayNameGOC; // CString is a MFC specific class
BSTR  bszTemp;

…
// pCI is a pointer to the IRequestSDK interface
pCI->get_DisplayNameGOC(&bszTemp);

// Get the project name into a CString object
fromBSTR(bszTemp, szDisplayNameGOC);
}
…
void fromBSTR(BSTR bszStr, CString & szValue)
{
                szValue = bszStr;
}
```

3.30 put_DisplayNameGOC

put_DisplayNameGOC(BSTR *newVal)

This function overwrites the geodes's display name text moniker used within the FoamSubApplicationClass instantiation. The display name text moniker is found with the //{{DISPLAYNAME-GOC keyword, in the .GOC file. The display name is the string found under the geode's main status pane icon (situated at the leftmost part of the screen). The text moniker, which contains the string, is defined elsewhere in the .GOC file (refer to the Set_Block code example to know how to changes that string from the tool – or refer to the /* {{ICON LABEL BLOCK}} -- DO NOT ERASE */ keyword in the .GOC file).

The following example demonstrates the use of the put_DisplayNameGOC method.

```
CString szDisplayNameGOC; // CString is a MFC specific class
BSTR  bszTemp;

…
// Change Display Name text moniker used in the geode's FoamSubApplicationClass for the NewTextMoniker
szDisplayNameGOC = "@NewTextMoniker";

bszTemp = toBSTR(szDisplayNameGOC);

// pCI is a pointer to the IRequestSDK interface
pCI->put_DisplayNameGOC(&bszTemp);


…

BSTR MainToolDialog::toBSTR(CString szValue)
{
        return szValue.AllocSysString();

}
```

3.31 get_ResourcesGOC

get_ResourcesGOC(BSTR *newVal)

This function reads all of the geode's resource definitions. Resources definitions are source code lines starting with the string "@start". This function can be used to scan the resources names already defined for that specific geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_ResourcesGOC method.

```
BSTR bszTemp;
CStringArray m_csaResourcesGOC;
CString szResourcesGOC;

…

// Initialize the string array
m_csaResourcesGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ResourcesGOC (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaResourcesGOC);

// Get all strings into one single string szResourcesGOC:
szResourcesGOC = fromArraytoStr(m_csaResourcesGOC;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.32 get_ObjectsGOC

get_ObjectsGOC(BSTR *newVal)

This function reads all of the geode's objects definitions. Objects definitions are source code lines starting with the string "@object". This function can be used to scan the object names already defined for that specific geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_ObjectsGOC method.

```
BSTR bszTemp;
CStringArray m_csaObjectsGOC;
CString szObjectsGOC;

…

// Initialize the string array
m_csaObjectsGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_ObjectsGOC (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaObjectsGOC);

// Get all strings into one single string szObjectsGOC:
szObjectsGOC = fromArraytoStr(m_csaObjectsGOC;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.33 get_MethodsGOC

get_MethodsGOC(BSTR *newVal)

This function reads all of the geode's method definitions. Method definitions are source code lines starting with the string "@method". This function can be used to scan the method names already defined for that specific geode. The returned value points to an array of strings.

The following example demonstrates the use of the get_MethodsGOC method.

```
BSTR bszTemp;
CStringArray m_csaMethodsGOC;
CString szMethodsGOC;

…

// Initialize the string array
m_csaMethodsGOC.RemoveAll();

// pCI is a pointer to the IRequestSDK interface
pCI-> get_MethodsGOC (&bszTemp);

//Extract the string array from the BSTR parameter
fromBSTR(bszTemp, m_csaMethodsGOC);

// Get all strings into one single string szMethodsGOC:
szMethodsGOC = fromArraytoStr(m_csaMethodsGOC;);

…

CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.34 get_Version

get_Version(float *pVal)

This fonctions reads the Nokia 9000 Developers Environment's version number (example: 1.0).

The following example demonstrates the use of the get_Version method.

```
float fVersion;
…

// pCI is a pointer to the IRequestSDK interface
pCI->get_Version(&fVersion);

// Check if version number is smaller that 1.5
if (fVersion < 1.5)
{
    AfxMessageBox("Warning: This tool requires SDK version 1.5 and higher!\n");
}
```

3.35 get_MainWinHandle

get_MainWinHandle(short *pVal)

This functions gets the Nokia 9000 Developers Environment main window handle. This value is required by the tool that need to display its own windows within the N9KDE frame.

The following example demonstrates the use of the get_MainWinHandle method.

```
MainDialog* dlg;
short sMainWinHandle;
…

// pCI is a pointer to the IRequestSDK interface
pCI->getMainWinHandle(&sMainWinHandle)

dlg = new MainDialog((HWND) sMainWndHandle);
dlg->DoModal();

…
```

3.36 Get_Block

Get_Block(BSTR *szStart, BSTR *szEnd, BSTR *szContents);

This function reads a block of code that is found between 2 string delimiters (szStart and szEnd), in any of the three baseline files of the currently opened project (.GOC, .GOH and .GP). The result is returned through the szContents parameter. The returned value points to an array of strings.

```
BSTR bszStart,bszEnd,bszContents;
CString szContents;
CStringArray csa;

m_StartGetBlock = "/* {{ICON LABEL BLOCK}} -- DO NOT ERASE */"
m_StartGetBlock = "/* {{END ICON LABEL BLOCK}} -- DO NOT ERASE */"

bszStart = toBSTR(m_StartGetSetBlock);
bszEnd = toBSTR(m_EndGetSetBlock);

// pCI is a pointer to the IRequestSDK interface
pCI->GetBlock(&bszStart,&bszEnd,&bszContents);

fromBSTR(bszContents,csa);
m_EditGetSetBlock = fromArraytoStr(csa);
…


CString fromArraytoStr(CStringArray & csa)
{
        // concatenate the strings, delimited by "\n"
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                szTmp += csa.GetAt(i);
                szTmp += "\n";
        }
        return szTmp;
}


BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )
                {
                        szTmp += "\n";
                }
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();

}

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

3.37 Set_Block

Set_Block(BSTR *szStart, BSTR *szEnd, BSTR *szContents);

This function overwrites a block of code that is found between 2 string delimiters (szStart and szEnd), in any of the three baseline files of the currently opened project (.GOC, .GOH and .GP). The szContents parameter must point to the string array containing the code block to write.

```
BSTR bszStart,bszEnd,bszContents;
CString szContents,  szNewTitle;
CStringArray csa;
int i;

m_StartGetBlock = "/* {{ICON LABEL BLOCK}} -- DO NOT ERASE */"
m_StartGetBlock = "/* {{END ICON LABEL BLOCK}} -- DO NOT ERASE */"
szNewTitleMoniker = ""@visMoniker NewTextMoniker =\"New App\"";

bszStart = toBSTR(m_StartGetSetBlock);
bszEnd = toBSTR(m_EndGetSetBlock);

// pCI is a pointer to the IRequestSDK interface
pCI->GetBlock(&bszStart,&bszEnd,&bszContents);

fromBSTR(bszContents,csa);

// find visMoniker string in order to replace it
while (i < csa.GetSize())
{
        if (csa[i].Find("@visMoniker") != -1)
                break;
        i++;
}

// Replace actual title with a new application title (status pane title)
csa.SetAt(i szNewTitleMoniker)

bszContents = toBSTR(csa);

pCI->SetBlock(&bszStart,&bszEnd,&bszContents);
…

BSTR MainToolDialog::toBSTR(CString szValue)
{
        return szValue.AllocSysString();

}

BSTR toBSTR(CStringArray & csa)
{
        CString szTmp;
        for( int i = 0; i < csa.GetSize(); i++ )
        {
                if( i )
                {
                        szTmp += "\n";
                }
                szTmp += csa.GetAt(i);
        }
        return szTmp.AllocSysString();

}

void fromBSTR(BSTR bszStr, CStringArray & csa)
{
        CString szTmp = bszStr;
        CString szRight, szLeft;
        int nLoc = -1;
        int i = 0;
        while((nLoc = szTmp.Find( '\n' )) >= 0 && i < 10240)
        {
                if( nLoc )      szLeft = szTmp.Left( nLoc );
                else szLeft = "";

                csa.Add( szLeft );
                szTmp = szTmp.Mid( nLoc + 1);
                i++;
        }
        csa.Add( szTmp );
}
```

4 Glossary

ATL	ActiveX Template Library
COM	Component Object Model
GUID	Globally Unique Identifier
IID	Interface Identifier
N9KDE	Nokia 9000/9110 Development Environment
SDK	Software Development Kit

5 References

ATL Programming

Professional COM Applications with ATL
Sing. Li, Panos. Economopoulos, , WROX Press Inc., 1998, ISBN 1-861001-7-03