Q211802: HOWTO: Query MSI tables at Runtime

Products

InstallShield 2009 Premier, InstallShield 2009 Professional, InstallShield 2010 Premier, InstallShield 2010 Professional, InstallShield 2011 Premier, InstallShield 2011 Professional

Project Type

Basic MSI

Summary

If you are looking to query the MSI tables of your install during runtime this can be done with Windows Installer Database Functions. This article demonstrates how to access MSI tables during runtime.

Discussion

Microsoft offers a set of Windows Installer Database Functions that can be used within InstallScript. To create a select statement, you'll need to create a view. This can be done with the MsiDatabaseOpenView(MSIHANDLE hDatabase, LPCTSTR szQuery, MSIHANDLE *phView) function.

Parameters
hDatabase [in]
Handle to the database to which you want to open a view object. You can get the handle as described in Obtaining a Database Handle.
szQuery [in]
Specifies a SQL query string for querying the database. For correct syntax, see SQL Syntax.
phView [out]
Pointer to a handle for the returned view.

One way to find hDatabase is:

```
hDatabase = MsiGetActiveDatabase(ISMSI_HANDLE );
```

Once you have phView as the return view, you can execute the view with MsiViewExecute(MSIHANDLE hView, MSIHANDLE hRecord)

Parameters
hView [in]
Handle to the view upon which to execute the query.
hRecord [in]
Handle to a record that supplies the parameters. This parameter contains values to replace the parameter tokens in the SQL query. It is optional, so hRecord can be zero.

Next, you'll want to loop through rows to get the information you want from each row. This can be done with MsiViewFetch(MSIHANDLE hView, MSIHANDLE *phRecord)

Parameters
hView [in]
Handle to the view to fetch from.
phRecord [out]
Pointer to the handle for the fetched record.

Here's an example of the while loop.
while(MsiViewFetch(hView, phRecord) != ERROR_NO_MORE_ITEMS)

To get values of columns within the while loop you'll want to call MsiRecordGetString(MSIHANDLE hRecord, int iField, LPTSTR szValueBuf, DWORD *pcchValueBuf)

Parameters
hRecord [in]
Handle to the record.
iField [in]
Specifies the field requested.
szValueBuf [out]

Pointer to the buffer that receives the null terminated string containing the value of the record field. Do not attempt to determine the size of the buffer by passing in a null (value=0) for szValueBuf. You can get the size of the buffer by passing in an empty string (for example ""). The function then returns ERROR_MORE_DATA and pcchValueBuf contains the required buffer size in TCHARs, not including the terminating null character. On return of ERROR_SUCCESS, pcchValueBuf contains the number of TCHARs written to the buffer, not including the terminating null character.
pcchValueBuf [in, out]
Pointer to the variable that specifies the size, in TCHARs, of the buffer pointed to by the variable szValueBuf. When the function returns ERROR_SUCCESS, this variable contains the size of the data copied to szValueBuf, not including the terminating null character. If szValueBuf is not large enough, the function returns ERROR_MORE_DATA and stores the required size, not including the terminating null character, in the variable pointed to by pcchValueBuf.

Additional Information

Here's a link to a Microsoft site with more info on their functions:
http://msdn.microsoft.com/en-us/library/aa368250(v=VS.85).aspx

Here's some sample code that queries the Directory table and displays the first column of each row in the table at runtime.

```
function MyFunction(hMSI)

HWND hDatabase, hView, hRec;
   STRING svValue;
   NUMBER nvBufferSize, nResult;
begin

hDatabase = MsiGetActiveDatabase(ISMSI_HANDLE );
MsiDatabaseOpenView(hDatabase,"SELECT * FROM Directory", hView);
nResult = MsiViewExecute(hView,0);
while(MsiViewFetch(hView, hRec) != ERROR_NO_MORE_ITEMS )
nvBufferSize = 256;
MsiRecordGetString(hRec, 1, svValue, nvBufferSize);
SprintfBox(INFORMATION,"test","%s",svValue);
endwhile;
MsiCloseHandle(hRec);

end;
```

Last Modified Date: 01-24-2011 **ID:** Q211802