

Sample ODPP Affinity Language Exit



IBM's Optim Enterprise Solution

1.	Introduction.....	3
2.	Affinity Language Exit.....	3
3.	Inventory of Files	4
4.	Building the Exit.....	4
4.1	Windows	4
4.2	Linux	4

1. Introduction

The client may write an exit that defines a custom language. The exit name must have a prefix "ioqx0". The sample exit supplied with the ODPP binaries is named ioqx0affexit.dll / libioqx0affexit.so

2. Affinity Language Exit

The exit definitions are kept very simple and only standard, built-in C types are used.

Value	Name	Description
0	EXIT_RC_OK	The operation was successful.
1	EXIT_RC_INVALID_CLASS_COUNT	The number of custom classes defined exceed the maximum number of classes supported.
2	EXIT_RC_INTERNAL_ERROR	The operation was unsuccessful. An error condition arose within the exit preventing it from continuing.

The exit header file defines the return codes as follows:

```
#define EXIT_RC_OK 0
#define EXIT_RC_INVALID_CLASS_COUNT 1
#define EXIT_RC_INTERNAL_ERROR 2
```

The exit must be built as a dynamic library (DLL on Windows; shared library on UNIX/Linux) and must implement a public (non-static) function named GetLanguageCustom.

Since the exit is always dynamically loaded, pointers to the functions will be obtained at run-time by name - so the names are important and are case sensitive.

The exit header file defines the function types as such:

```
typedef int Exit_GetLanguageCustom(UChar32 *pBuf,
                                   int iBufSizeChars,
                                   short *psClassCount,
                                   short *Sizes);
```

The function must be declared prior to use in the exit:

```
Exit_GetLanguageCustom GetLanguageCustom;
```

Once declared, the function can now be implemented. The only changes required are to numberOfCharacterTypes and pCharacterSets. A code point may be defined by its hex value as shown in the sample.

```

int GetLanguageCustom(UChar32 *pBuf,
                    int iBufSizeChars,
                    short *psClassCount,
                    short *Sizes)
{
    short numberOfCharacterTypes = 3;
    ODPP_WCHAR *pCharacterSets[3] = { ODPPCNVTOWSTR1("abcdefghijklmnopqrstuvwxyz\x00e6\x00e5\x00f8"),
                                      ODPPCNVTOWSTR1("ABCUVWXYZ\x00c6\x00c5\x00d8"),
                                      ODPPCNVTOWSTR1("0123456789")
    };

    return(SetLanguage(pBuf, iBufSizeChars, psClassCount,
                     Sizes, pCharacterSets, numberOfCharacterTypes));
}

```

3. Inventory of Files

The following is an inventory of files needed to create an ODPP Affinity Language Exit. The files are located under the Samples\ODPP_AFFLANGEXIT folder.

File	Description
odpp_affinityexit.h	ODPP Affinity Language Exit main header. Contains the manifest constants and function prototypes used by an exit.
odpp_affinityexit.c	ODPP Affinity Language Exit Source module
IOQAffinityExitWinExp.def	Windows Module Definition file
ODPPAFFLANGEXIT.vcproj	VS2008 project
ODPPAFFLANGEXIT.sln	VS2008 solution
makewinafflangexit.BAT	Build script for Windows
makerhelafflangexit.bsh	Build script for RHEL
IOQAffinityExitUnixExp.def	Unix Module Definition file

4. Building the Exit

4.1 Windows

The Exit must be built with Visual Studio 2008.

To build the Exit on Windows do one of the following:

1. Set the VSROOT2K8 environment variable to the root of your Microsoft Visual Studio 2008 installation folder and run the makewinafflangexit.BAT batch/command file.

- or -

2. From Visual Studio 2008, load the ODPPAFFLANGEXIT.sln file and build the project using the Release Solution Configuration.

4.2 Linux

1. Ensure that the GCC environment variable in the script points to the location of the gcc compiler on your machine.

2. Create a directory on the RHEL machine and copy odpp_affinityexit.h (header), odpp_affinityexit.c (source code), makerhelafflangexit.bsh (rhel build script) to this directory.
3. Set permissions for the build script:
 `chmod 755 makerhelafflangexit.bsh`
4. Run the makerhelafflangexit.bsh as follows:
 `./makerhelafflangexit.bsh <exit-source-directory>`