Connect:Direct for Windows .Net SDK

Overview

The Connect:Direct for Windows .Net SDK allows system programmers to extend the capabilities of the Connect:Direct for Windows environment. This SDK supports any version of the .Net framework from Microsoft using any .Net supported programming language (C#, VB.Net, J#, etc.).

Connection Settings

The Connect:Direct for Windows .Net SDK can utilize pre-configured connection settings. For information on this feature see Editing Connections Settings in the *Connect:Direct for Windows SDK Programming Guide*.

Sample Programs

Several sample source code projects are provided. These are meant to show basic examples of how to use the .Net SDK. To run the samples, you will need CdCore.dll and ConnectDirectSdk.dll in your executable path. You can copy these files to the same directory as the sample executables.

VB.Net

A Vb.Net sample program, VbDotNetSample1, has been provided. This is a console program that connects to a node, submits a process from a file, and displays the statistics for the process submitted. Look for the "$todo" tags in the "Module1.vb" source file and change the variables to be valid for your Connect:Direct for Windows node.

C#.Net

Two C# sample GUI programs are provided. Look for the "$todo" tags in the "SDKInterface.cs" source files for each sample and change the variables to be valid for your Connect:Direct for Windows node.

DotNetSample1 – This program connects to a node, issues a select process, then displays the process information returned.

DotNetSample2 – This program connects to a node, submits a process and then displays the process information and the statistics information for the process submitted.

Using the .Net Class Interface

Files

The following files are provided with the Connect:Direct for Windows .Net SDK.

1) ConnectDirectSdk.dll – This Module is a managed dll which interfaces your .Net managed program to the unmanaged Connect:Direct for Windows dll (CdCore.dll). This file must be copied to a folder that is in your executable path.
2) CdCore.dll – This Module is an unmanaged dll which interfaces to the Connect:Direct for Windows Server. This file must be copied to a folder that is in your executable path.
3) ConnectDirectSdk.xml – This module is the Intellisense help file which Visual Studio uses for quick information, auto completion and parameter help.

Adding to your .Net project

Project Properties

To use the Connect:Direct for Windows .Net SDK interface you must add the ConnectDirectSdk.dll as a reference in your Visual Studio project. Do the following steps:

1) Select "Project" from the menu then select "Add Reference"
2) Select "Browse" to search for the ConnectDirectSdk.dll
3) Browse to the location of the Connect:Direct for Windows .Net SDK installation. The default location is "C:\Program Files\Sterling Commerce\Connect Direct v4.5.00\SDK.Net\Sdk_Files\Release"
4) Click (highlight) the file "ConnectDirectSdk.dll" then click "OK"

Source Modules

Also, within your source file, you must import the ConnectDirectSdk namespace.

In Vb.Net, add the following to your source modules:

```
"Imports ConnectDirectSdk"
```

In C#.Net, add the following to your source modules:

```
"Using ConnectDirectSdk;"
```

Classes

Node

The Node class is the main interface to the Connect:Direct for Windows Server. It contains the high-level Connect:Direct functionality. It allows you to connect, submit processes, select statistics, etc. Almost all access to the Connect:Direct for Windows Server is through the Node object. The Node object creates and removes the connection to the Connect:Direct for Windows Server. Connections are shared and reused as different requests are made.

Process

The Process class allows you to retrieve information about processes that you submit or that are in the TCQ. It contains all of the process criteria returned from a Submit or SelectProc method call.

Statistic

The Statistic class allows you to retrieve the statistic records from the TCQ. It represents a group of records in the statistics database. They are returned from a SelectStat method call.

Node Class

Connecting to the Connect:Direct for Windows Node

The name of the Connect:Direct node and the connection information is set at object creation time using the Node constructor. If a parameter is not supplied (NULL pointer), the default value for that parameter is read from the Registry. During construction, the Node object attempts to connect to the physical Connect:Direct node, using the protocol information contained in the Registry. If the connection fails, an exception is thrown.

In the following constructor, stNode is required. stUser and stPass are optional. stPass is ignored if stUser is not provided.

```
Node(String stNode, String stUser, String stPass)
```

In the following constructor, stLcuFile is required. This is the file spec for an LCU file that contains the login information.

```
Node(String stLcuFile)
```

Disconnecting Nodes

Use the “DisconnectAll” method to disconnect from all Nodes.

```
bool DisconnectAll()
```

Submitting Processes

Use the “Submit” and “SubmitFile” to submit processes for execution on the node. This automatically creates a Process object and associates it with the node for the Submit.

This is the standard “SubmitFile” method. “stFileName” is required and is the file spec of the Connect:Direct process to submit.

```
void SubmitFile(String stFileName)
```

This “SubmitFile” method allows more control of the submitted process. “stFileName” is required and is the file spec of the Connect:Direct process to submit. “holdOverride” allows you to place this process in the Hold queue. “startTime” allows you to specify the time to run the process. “symbolics” are the substitution paramaters to apply to this process.

```
void SubmitFile(String stFileName, Hold holdOverride, String startTime,
                          Dictionary<String, String> symbolics)
```

The “Submit” method is very similar to the “SubmitFile” method above but instead of passing a file name of the process to submit, you pass “stText” which is the text itself of a Connect:Direct process to submit.

```
void Submit(String stText, Hold holdOverride, String startTime,
                          Dictionary<String, String> symbolics)
```

Manipulating Processes

The Node object has several methods that allow you to manage processes. You can view, change and delete processes. You can also place a process on Hold and release it from Hold. Each of these methods will return process information in the “ProcessList” property of the Node class of each process that was selected or changed.

This “SelectProc” method allows you to retrieve a list of all of the processes from the TCQ.

```
void SelectProc()
```

This “SelectProc” method allows you to retrieve a list of all of the processes from the TCQ whose process name matches “stName”.

```
void SelectProc(String^ stName)
```

This “SelectProc” method allows you to retrieve a list of all of the processes from the TCQ whose process number matches “nNumber”.

```
void SelectProc(int nNumber)
```

This “SelectProc” method allows you to retrieve a list of all of the processes from the TCQ whose process name matches any name in the array “arrayNames”.

```
void SelectProc(array<String^>^ arrayNames)
```

This “SelectProc” method allows you to retrieve a list of all of the processes from the TCQ whose process number matches any number in the array “arrayNumbers”.

```
void SelectProc(array<int>^ arrayNumbers)
```

The “HoldProc” method allows you to place a process in the TCQ on HOLD. “pProcess” is a process object.

```
void HoldProc(Process^ pProcess)
```

This “ReleaseProc” method allows you to release a process in the TCQ to run that was on HOLD. “pProcess” is a process object.

```
void ReleaseProc(Process^ pProcess)
```

This “ReleaseProc” method allows you to release a process in the TCQ to run that was on HOLD. “nNumber” is the process number of the process to release, “stPNode” is the primary node of the processes to release, and “stUserid” is the Userid of the processes to release.

```
void ReleaseProc(int nNumber, String^ stPNode, String^ stUserid)
```

This “DeleteProc” method allows you to delete a process from the TCQ. “pProcess” is a process object.

```
void DeleteProc(Process^ pProcess)
```

This “DeleteProc” method allows you to deleta a process from the TCQ. “nNumber” is the process number of the process to delete, “stPNode” is the primary node of the processes to delete, and “stUserid” is the Userid of the processes to delete.

```
void DeleteProc(int nNumber, String^ stPNode, String^ stUserid)
```

Retrieving Statistics

Use the “SelectStat” methods to retrieve statistics from the stats database. Stats are returned in the “StatsList” property of the Node class.

This “SelectStat” method allows you to retrieve a list of all statistic records from the stats database. Please note that this could be very large depending on how many days of stats records are kept in the database.

```
void SelectStat()
```

This “SelectStat” method allows you to retrieve a list of all statistic records from the stats database for a specific process. “pProcess” is the process object to retrieve the stats for.

```
void SelectStat(Process^ pProcess)
```

This “SelectStat” method allows you to retrieve a list of all statistic records from the stats database within a specified time range. “dtBegin” is the beginning timestamp for the stats and “dtEnd” is the ending timestamp. The timestamps are in the format of “MM/DD/YYYY hh:mm:ss AM|PM”.

```
void SelectStat(String^ dtBegin, String^ dtEnd)
```

Node Properties

ApiVersion	Returns the CD API version of the node as a long.
CDName	Returns the Connect:Direct node name sent to the client after successfully logging on to the node.
Name	Returns the alias node name passed in the constructor.
OSSubType	Returns the Operating System sub-type (additional information) of the Node.
OSType	Returns the Operating System type of the Node.
ProcessEntry	Returns the process from a Submit call.
ProcessList	Returns an array of processes.
SecurePlusSupported	Indicates whether the node supports Secure Plus.
SecurePlusVersion	Returns the Secure Plus version information as a long
Server	Returns the file server name the Connect:Direct node is running on.
StatsList	Returns an array of Stat messages from a SelectStat call.
Userid	Returns the User ID used to log into the node.

Process class

The process class contains all of the process criteria returned from a SUBMIT or SELECT PROCESS method. Processes are submitted using the Node.Submit or Node.SubmitFile methods.

Waiting for process completion

This "WaitForCompletion" method blocks the current thread until the process has exited all queues on the Connect:Direct Server, including error queues. It waits indefinitely.

```
void WaitForCompletion()
```

This "WaitForCompletion" method blocks the current thread until the process has exited all queues on the Connect:Direct server, including error queues, or until the timeout period has expired. "timeout" is in milliseconds.

```
void WaitForCompletion(long timeout)
```

Process properties

ByteCount	Returns the Bytes read/written from the file as a long.
Checkpoint	Returns the Checkpointing Enabled flag.
Class	Returns the session class property as a String.
ConditionCode	Returns the Return Code as an int.
DestDisp1	Returns the Destination Displacement 1 as a char.
DestDisp2	Returns the Destination Displacement 2 as a char.
DestDisp3	Returns the Destination Displacement 3 as a char.
DestFile	Returns the Destination File Name as a String.
ExecPriority	Returns the Current Execution Priority as a String.
ExtendedCompression	Returns the Extended Compression flag.
Feedback	Returns the Additional Return Code Information as an int.

FromNode	Returns the From Node flag.
Function	Returns the Current Function Executing as a String.
Hold	Returns the Hold flag as a char.
LocalNode	Returns the Local Node indicator flag.
LogDateTime	Returns the Logged Timestamp.
MsgData	Returns the Message Substitution Data as a String.
MsgId	Returns the Message Identifier field as a String.
MsgText	Returns the Message Text field as a String.
Name	Returns the Process Name as a String.
Number	Returns the Process Number as an int.
PNode	Returns the Primary Node Name as a String.
Priority	Returns the Current Priority as in int.
Queue	Returns the Process Queue as a String.
RecordCount	Returns the Records read/written as a long.
Restart	Returns the Restart flag.
Retain	Returns the Retain flag as a char.
SchedDateTime	Returns the Scheduled Timestamp.
SecureEnabled	Returns the Secure+ enabled flag.
SecureProtocol	Returns the Secure+ Protocol as a String.
Signature	Returns the Secure+ effective Signature setting.
SNode	Returns the Secondary Node Name as a String.
SourceDisp1	Returns the Source Displacement 1 as a char.
SourceDisp2	Returns the Source Displacement 2 as a char.
SourceDisp3	Returns the Source Displacement 3 as a char.

SourceFile	Returns the Source File Name as a String.
SSLCipherSuite	Returns the Secure+ SSL Cipher Suite as a String.
StandardCompression	Returns the Standard Compression flag.
Status	Returns the Current Status as a String.
StepName	Returns the Current Stepname as a String.
STSCtrlCipher	Returns the Secure+ STS Control Cipher as a String.
STSDataCipher	Returns the Secure+ STS Data Cipher as a String.
SubmitNode	Returns the Submitter's Node Name as a String.
Submitter	Returns the Submitter's User Id as a String.
XmitBytes	Returns the Bytes sent/received count as a long.
XmitRUs	Returns the RUs sent/received as a long.

Statistic Class

The Statistic class represents a group of records in the statistics database. They are returned by a "SelectStat" method call.

Audit Information

The "GetAuditField" method returns the value of the field requested from the Stats Audit Information. Audit data in Stats records is optional and Stat records can have different audit field available. "stField" is the name of the audit field you are requesting information for; "stValue" is the value of the field requested. This method returns TRUE if the audit field was found and FALSE if not.

```
BOOL GetAuditField(String^ stField, String^% stValue)
```

Statistic Properties

ConditionCode	Returns the Return Code.
Feedback	Returns the Additional Return Code information.

LogDateTime	Returns the Logged Timestamp.
MsgData	Returns the Message Substitution Data as a String.
MsgId	Returns the Message Identifier field as a String.
MsgText	Returns the Message Text field as a String.
ProcessName	Returns the Name of the process.
ProcessNumber	Returns the Process number.
RecCat	Returns the Record Category.
RecId	Returns the Record Identifier tag.
SNode	Returns the Secondary Node Name.
StartDateTime	Returns the Start Timestamp.
StopDateTime	Returns the Stop Timestamp.
Submitter	Returns the User Id of the submitter.