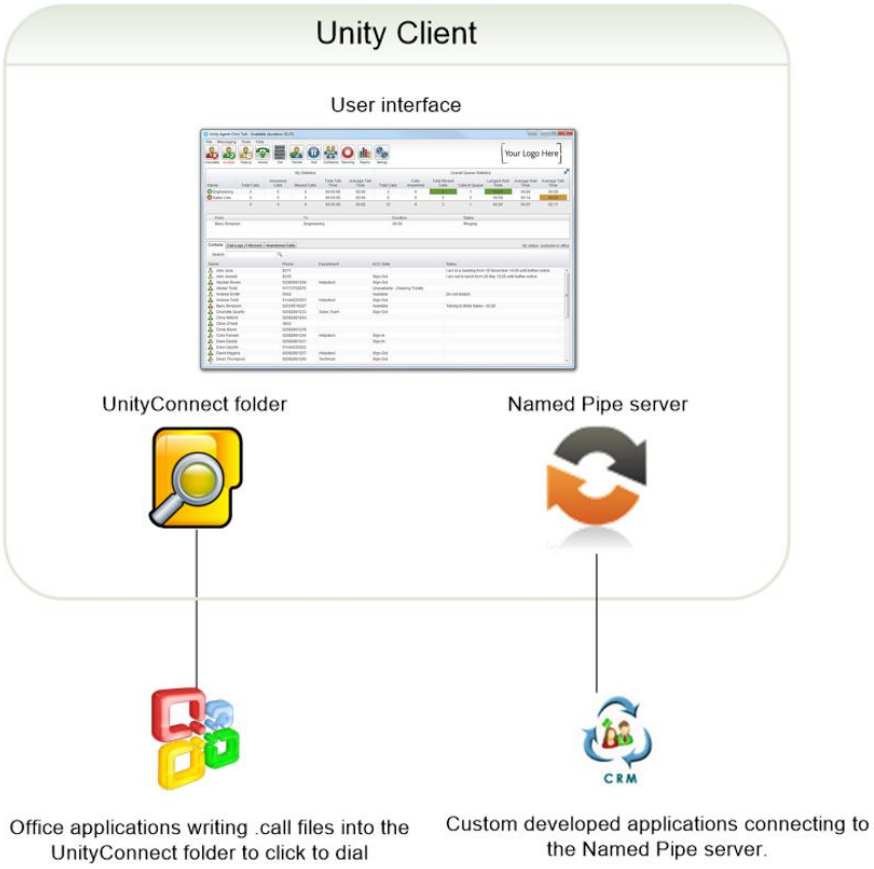
KAKAPO SYSTEMS

UNITY CONNECT API OVERVIEW

INTRODUCTION

Unity Connect is an API that offers full control and presence management to third-party applications through the Unity Client. Integration can be achieved either through the file system or a communications technology called Named Pipes, both of which are explained below. Multiple applications can consume Unity Connect functionality simultaneously to provide call control from any application running on the desktop.



All communication between the third-party application and the Broadworks core is proxied through the Unity client. This ensures that as the Broadworks platform moves from release to release, the integration between Unity and the third party application remains valid, eliminating the need to continually manage [and update] code. Because Unity is an automatically updating application, the link between the third party application and Broadworks will remain intact regardless of the Broadworks version being used.

LICENSING

Unity Connect relies on the Unity Desktop/Agent/Supervisor Enterprise license being assigned. Please speak to your service provider about assigning this license. Unity Connect is also available with trial licenses.

USING THE FILE SYSTEM FOR CLICK TO DIAL

The easiest way to integrate with Unity Connect is through the file system, although only basic dial functionality is available. When Unity starts (assuming that an enterprise or trail license is assigned) a UnityConnect folder will be automatically created in the installation folder, by default C:\Program Files\Unity Client. When a file with the .call file extension is placed or created in this folder, UnityConnect will automatically read the file, make the call and delete the file. For example if a file named 00442082881248.call is placed in this folder Unity will instantly dial 00442082881248. The "+" character will automatically be replaced with "00", so +442082881248.call is a valid filename.

To achieve basic click-to-dial using this mechanism, simply create a script that automatically places files in this folder using the above file naming scheme. Unity will then place the call and delete the file. If the user is already on a call, it will be placed on hold and the new call made.

The following code sample illustrates how easy it is to add click to dial functionality to an application developed in Visual Basic 6 (excluding error handling).

```
Public Sub DialNumberWithUnity(number As String)

On Error Resume Next

If number <> "" Then

        Open "C:\Program Files\Unity Client\UnityConnect\" + number + ".call" For Output As #1

        Close #1

End If

End Sub
```

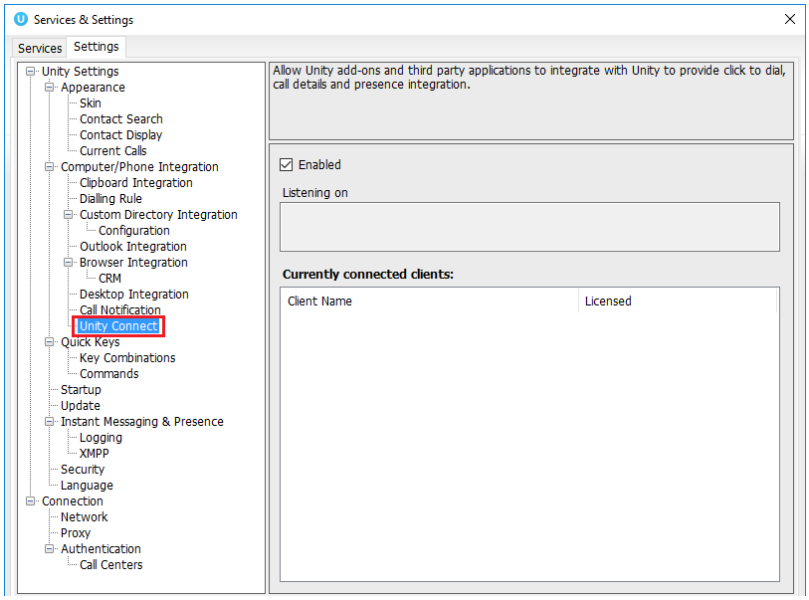
When running Unity in a terminal services or Citrix environment a new folder is created for every user, which is based on the formatted VoIP login id without the domain. For example if user chris.tutt@kakaposystems.com is running Unity in a terminal services/Citrix environment, a folder called christutt will be created within the installation directory and the UnityConnect folder will be placed within that folder. This ensures that when .call files are placed in these folders the correct Unity instance makes the call. This must be considered when writing the script to create the file in a UnityConnect folder. You can check the folder using Windows Explorer to ensure the file is created in the correct folder.

USING THE NAMED PIPES SERVER

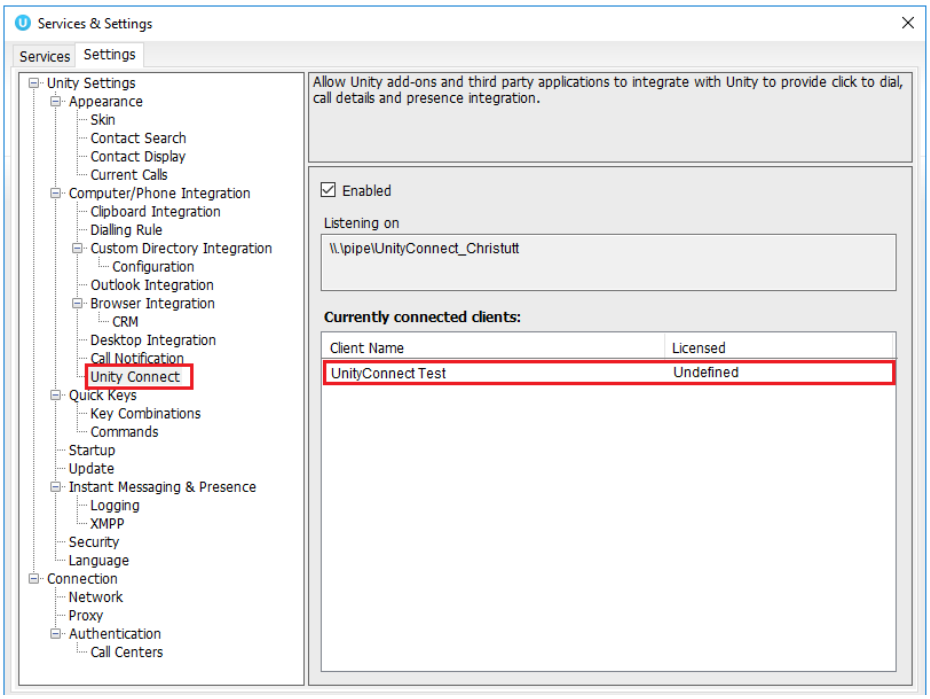
This method is more complicated but offers complete call control including dial, release, hold, and transfer (both with and without announce). In order to use names pipes you must have access to the code of the third party system you wish to integrate with Unity Connect, as well as development experience with technologies such as XML, networking and named pipes.

It should be noted that the named pipe server is built into the Unity application; no separate installation required is required.

Once Unity is successfully licensed a named pipe server instance is started (assuming an enterprise or trial license is assigned) based on the formatted VoIP login id without the domain. For example if the VoIP login id is chris.tutt@kakaposystems.com then the pipe will be \\.\pipe\UnityConnect_christutt, as shown below.



This ensures that in a terminal services/Citrix environment [where all users are consuming the same instance of Unity Client] each user is uniquely identified. When connecting to the named pipe server the correct pipe name must be specified otherwise the connection attempt will fail. You can check the name of the pipe by selecting Settings > Settings tab > Unity Connect. If your third party application has successfully connected through Unity Connect it will be displayed in the list, as shown below.



All third-party applications are assigned an undefined license once connected through Unity Connect, while some Unity add-ins (such as the Unity TAPI driver) may require a separate license. This is a future development phase and is outside the scope of this document. As long as the license type is either true or undefined communication will be permitted.

Once the third party has successfully connected to UnityConnect it should identify itself so that the name is included in the "currently connected clients" list above. There is no limit to the length of the name, however the recommended maximum length is 30 characters to ensure it is shown correctly in the list.

```
<UnityConnect version='1.0'><Command Action='SetName'><Name>UnityConnect Test</Name></Command></UnityConnect>
```

NAMED PIPE MESSAGING FOR CALL CONTROL

All messaging between the third-party application and Unity Connect is in basic XML format, minus XML declaration elements.

There is a sample application written in C# which illustrates how to implement this functionality. Please note the application is provided "as-is" and should only be used in a development environment as no guarantee is offered. The sample application does not include complete error handling or logic; hence it is only to be used as a reference to outline the communication between the third party application and the Unity Connect named pipes server. A short guide to the sample application is given in Appendix 2, below.

DIAL

This message is sent from the third party application to Unity Connect to place a call. Any illegal characters included in the number (such as "+" or "-") are removed. The number value can be either a valid extension or phone number.

```
<UnityConnect version='1.0'><Command Action='Call'><Number>+44-20828821248</Number></Command></UnityConnect>
```

CALL UPDATE

This message is sent from Unity Connect to the third-party application whenever information is received regarding the call. The Call id is unique to the call (if there are multiple calls in place) and the status can be any of the following values:

Call Status	Description
Alerting	The call is ringing
Active	The call has been answered or taken off hold
Hold	The call is on hold
Released	The call has ended
Detached	The call has ended

```
<UnityConnect version='1.0'><Command Action='CallUpdate'><CallId>localHost561288037:0</CallId><Status>Alerting</Status><RemoteName>Steve Tutt</RemoteName><RedirectName></RedirectName><RemoteNumber>1251</RemoteNumber></Command></UnityConnect>
```

RELEASE

This message is sent from the third party application to Unity Connect to hang up an existing call. The call id would have been reveived from the CallUpdate message. If the call has already been released this message is ignored.

```
<UnityConnect version='1.0'><Command Action='Release'><CallId>localHost561288037:0</CallId></Command></UnityConnect>
```

ANSWER

This message is sent from the third party application to Unity Connect to answer an inbound call that is ringing (alerting state). The call id would have been received from the CallUpdate message. If the call is already active this message is ignored.

```
<UnityConnect version='1.0'><Command Action='Answer'><CallId>
localHost561288037:0</CallId></Command></UnityConnect>
```

HOLD

This message is sent from the third party application to Unity Connect to put an active call on hold. The call id would have been received from the CallUpdate message. If the call is already on hold or the call is alerting (meaning it hasn't been answered yet) then this message is ignored.

```
<UnityConnect version='1.0'><Command Action='Hold'><CallId>
localHost561288037:0</CallId></Command></UnityConnect>
```

RETRIEVE

This message is sent from the third party application to Unity Connect to retrieve a held call. The call id would have been received from the CallUpdate message. If the call is not held then the message is ignored.

```
<UnityConnect version='1.0'><Command Action='Retrieve'><CallId>
localHost561288037:0</CallId></Command></UnityConnect>
```

BLIND TRANSFER

This message is sent from the third party application to Unity Connect to tranfer the call to a third party number. The call id would have been received from the CallUpdate message. The call can either be active or held, otherwise the message is ignored. Any illegal characters included in the number (such as "+" or "-") are removed. The number value can be either a valid extension or phone number.

```
<UnityConnect version='1.0'><Command Action='Transfer'><CallId>
localHost561288037:0</CallId><Number>1211</Number></Command></UnityConnect>
```

TRANSFER WITH ANNOUNCE

This message is sent from the third party application to Unity Connect to transfer two calls together. The call id of both calls would have been received from CallUpdate messages. Either call can be active or held, otherwise the message is ignored.

```
<UnityConnect version='1.0'><Command
Action='TransferConsult'><CallId>localHost561288037:0</CallId><CallId>
localHost561288556:1</CallId></Command></UnityConnect>
```

APPENDIX 1: FINDING THE FORMATTED BROADWORKS ID

To find the formatted Broadworks ID which is used when using the Unity Connect named pipes server or when using the file system mechanism in a terminal services/Citrix environment, use the following rules.

Starting with +44-2082881248@kakaposystems.com

1] Remove the domain, leaving +44-2082882128

2] Remove all instances of "+", leaving 44-2082881248

3] Remove all instances of "-", leaving 442082881248

4] Remove all instances of ".", leaving 442082881248

5] Remove all instances of "_", leaving 442082881248

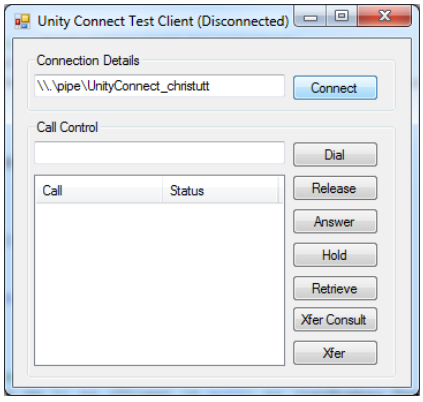
Thus the folder path will be C:\Program Files\Unity Client\442082881248\UnityConnect\ in a terminal services or Citrix environment and the named pipe name will be \\.\pipe\UnityConnect_442082881248 The sample application illustrates how to get the login id from the registry (where it is stored by Unity) and perform the required formatting.

APPENDIX 2: USING THE SAMPLE APPLICATION

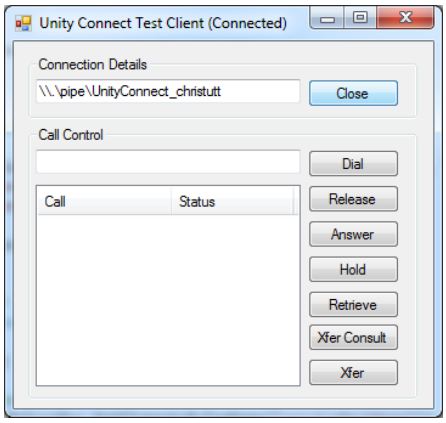
The sample application can be downloaded from
https://portal.unityclient.com/UpdateFiles/UnityConnectSampleApplication.zip

As mentioned the sample application is written in Visual Studio 2005 (.NET Framework 2.0) to illustrate the messaging between the third-party application and Unity Connect. This application is not designed to showcase best development techniques or standards, so should not be used in a production environment. Please study the messaging between the sample application and Unity to gain a thorough understanding of the UnityConnect XML document schema.

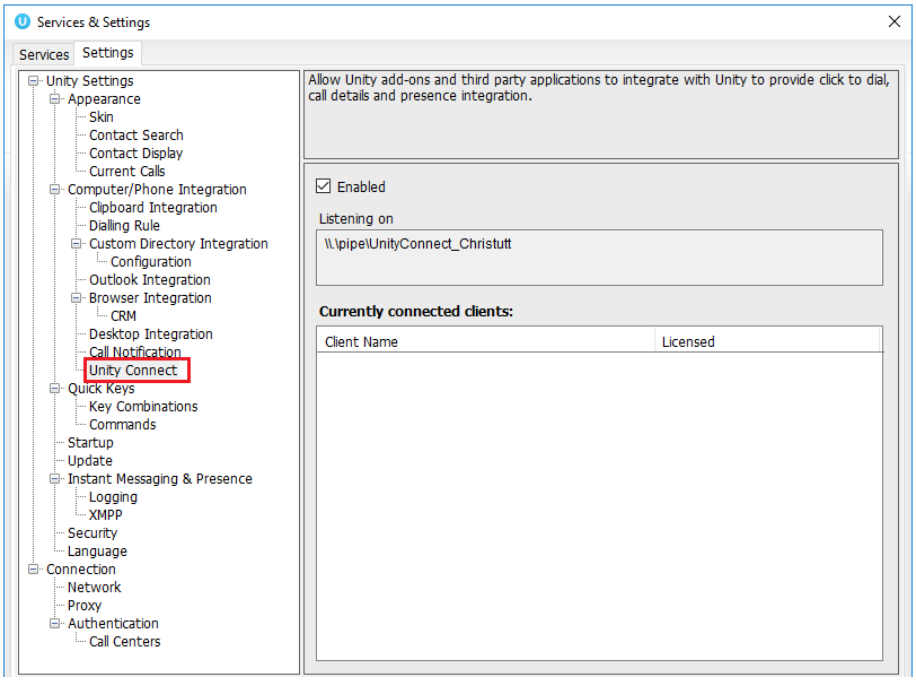
When the sample client starts you will see the below screen.



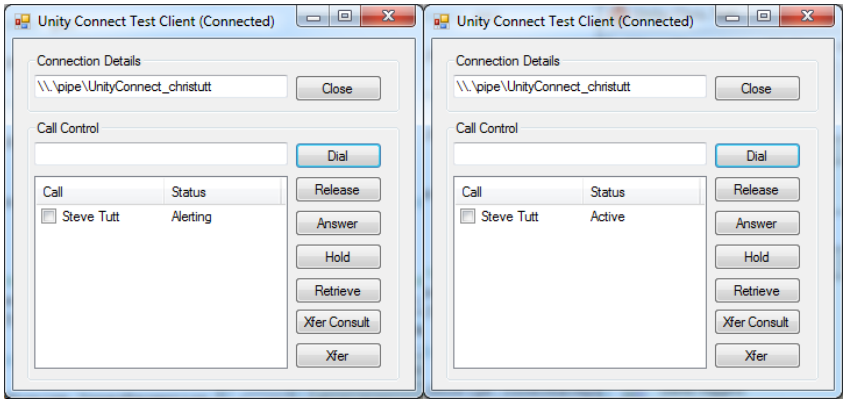
Check that the named pipe is correct and click Connect. If the connection attempt is successful you will see this in the title bar, as shown below



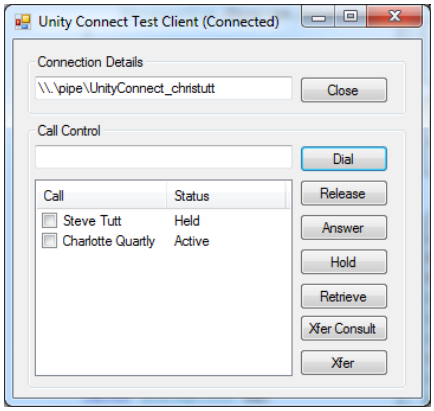
Otherwise an error message will be displayed if the connection attempt failed. The most likely reason for the connection attempt to fail is that the pipe isn't given the connect name or the named server isn't enabled in Unity. In either case go to Settings in Unity and check that UnityConnect is enabled, then ensure that the named pipe in the sample client is the same as the named pipe that Unity Connect is listening on.



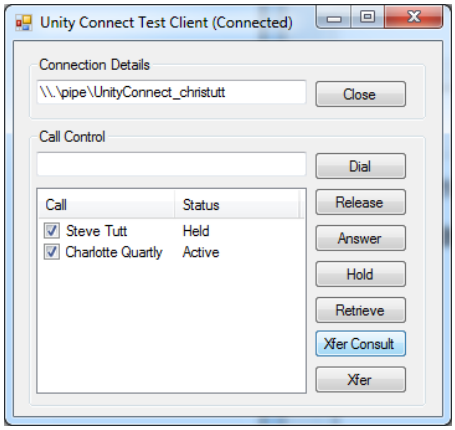
To dial a number, enter the number into the textbox next to the Dial button, then click Dial. You will see the call appear as "alerting", then once the remote party answers it will change to "active"



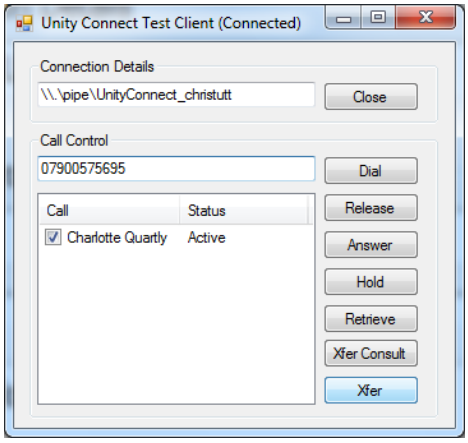
The call list shows all calls that the user is currently managing. The user must select the call [or calls] that they want to perform the action against.



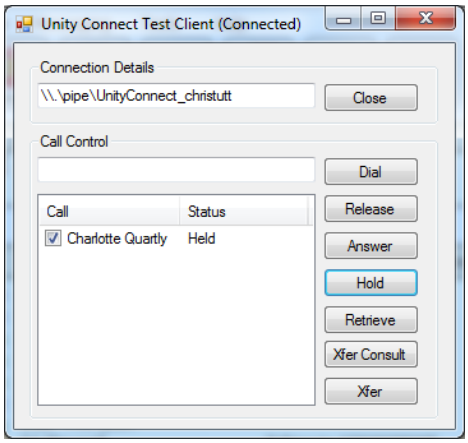
In the above illustration, I was on the phone to Steve Tutt. I then entered the extn for Charlotte into the textbox and clicked dial. The first call was automatically put on hold and the second call was established. If I then want to transfer the two calls together I select them both and click Xfer Consult, as shown below.



Similarly, if I have one call in the list that I want to transfer to a third party, I select the call, enter the number of the third party into the textbox, and click the Xfer button. This is illustrated below.



I can also hold or retrieve a call by selecting it from the list and clicking the appropriate button.



Lastly I can answer incoming calls by selecting it in the call list and clicking Answer, as illustrated below.