Phone Manager API Reference

MARCH 2017

DOCUMENT RELEASE 5.0

API REFERENCE

Table of Contents

NOTICE

The information contained in this document is believed to be accurate in all respects but is not warranted by Mitel Networks™ Corporation (MITEL®). The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes.

TRADEMARKS

Mitel Phone Manager API Reference
Release 5.0 - March, 2017

2 Introduction

Phone Manager allows for control and integration of the associated extension through the use of the Phone Manager API. There are several different components available including and these components can then be embedded within 3rd party applications, web pages etc. to control the extension and receive events associated with the extension.

A *Professional* license or higher is required to use some parts of the API

Each of the integration component options has different application requirements and restrictions depending on the scenario that they will be integrated into.

Component	*Commands*	*Events*	*Require Phone Manager to be running?*	*Requires Professional License?*
Microsoft .Net Framework DLL	✓	✓	✗	✓
Microsoft .Net Framework WCF	✓	✓	✓	✓
COM Component	✓	✓	✓	✓
Command Line Executable	✓	✗	✓	✓
Macro Scripting (VBScript)	✓	✓	✓	✓
TAPI (1st Party)	✓	✓	✓	✓
Contact Imports	✗	✓	✓	✗
Highlight and Dial	✓	✗	✓	✗
URI Schemes	✓	✗	✓	✗

3 Microsoft .Net Framework DLL

The .Net DLL is a Microsoft .Net Framework v4.0 assembly that can be used within a .Net development environment, for example VB.Net and C#. This is designed to replace the Phone Manager client as it does not require Phone Manager to be running on each user's desktop - but they can both be running at the same time. The 3rd party application manages the connection and receives all the events that happen on the extension and has access to call control features.

3.1 Getting Started

The .Net DLL assembly files are installed within the Phone Manager installation folder and so Phone Manager needs to be installed as this contains all the files that are required.

The following assemblies should be referenced from within your project:

- Xarios.PhoneManager.dll
- Xarios.PhoneManager.Types.dll
- Xarios.CallRecorder.Common.dll
- Xarios.CallRecorder.WCF.Types.dll

There are other assemblies that will need to be located in the same folder as the application executable, see the Deployment section for details.

> The target framework for the project needs to be set to .Net Framework 4 or higher and the platform needs to be set to x86.

See the "Phone Manager API Reference.chm" for more details.

Phone Manager requires that a session is created to the Communication Service before a CTI connection can be made. To create a session the *User* must have a valid account on the Communication Service that has the *Phone Manager* role and has at least a *Professional* license assigned. This license will allow one additional API connection for each *User,* i.e. if you run the full Phone Manager client AND an application using the .Net DLL then only one license will be consumed.

To create a session you need to provide:

- Discovery Server: This is the IP address or hostname of the Communication Server
- Client Credential Type: What authentication mechanism is used to validate the *User.* This is either *Windows* or *Basic* authentication.
- Username, Password, Domain: If provided then these will override the details of the current logged in *User.*

The C# code shows a how to create a session using *Windows* authentication.

```
PhoneManagerSession session = new PhoneManagerSession("cs_server");
session.SessionChanged += phoneSession_SessionChanged;
session.CreateSession();
```

The C# code shows a how to create a session using *Basic* authentication. When using *Basic* authentication then the *clientUsername* and *clientPassword* field are required. (Leave the *clientDomain* blank).

```
PhoneManagerSession session = new PhoneManagerSession("cs_server",
HttpClientCredentialType.Basic, "myusername", "mypassword", "");
session.SessionChanged += phoneSession_SessionChanged;
session.CreateSession();
```

The `session.CreateSession` method will return if the session has been created or throw an exception with a reason if this fails. The `SessionChanged` event

Once the session has been created then the CTI connection can be established to provide access to the methods and events from the PBX. An extension number is required to create the connection . The *User* on the Communication Server can have extensions associated with the account and these can be used or you could prompt the *User* for the extension to use.

```
PhoneManagerCTL phoneManager = new PhoneManagerCTL(); phoneManager.Initialised += phoneManager_Initialised; phoneManager.ConnectionFailed += phoneManager_ConnectionFailed; phoneManager.Initialise(session.SessionID, extension, session.Host, session.Port, session.UserID);
```

The `Initialise` method will return immediately but the application then needs to wait for the `Initialised` event to be raised as this indicates that a successful connection has been established. If the connection fails then a `ConnectionFailed` event will be raised with a message to indicate the reason why.

3.2 Deployment

The requirements for installing the full Phone Manager client apply when deploying the .Net DLL. See the Phone Manager user guide for details.

Applications that use the .Net DLL will need to deploy several assemblies to ensure that it can function correctly. The Phone Manager client does not have to be installed but it can and be run side by side with the application.

These assemblies will need to be located in the same folder as the application executable.

Assembly Name
Xarios.PhoneManager.dll
Xarios.PhoneManager.Types.dll
Xarios.OAIConnection.dll
Xarios.CallRecorder.WCF.Types.dll
Xarios.CallRecorder.Common.dll
Xarios.Collections.dll
Xarios.Strings.dll
Xarios.Logging.dll
NLog.dll

The assemblies are backwards compatible within the same *major.minor* release. The file versions will change for each release but the assembly versions will only change with each *major.minor* version release. For example, if the application has used the assemblies from v4.0.2900:

File Version	Compatible	
v4.0.2901.1	Yes	
v4.1.1001.1	No - requires re compilation with the latest assemblies or use of assembly redirection	
v5.0.1001.1	No - requires re compilation with the latest assemblies or use of assembly redirection	

3.3 Upgrading from v3.0

If any integration has been performed using previous versions of Phone Manager using the .Net DLL then they can be updated to support the latest API. This will require code changes and recompilation against the current assemblies in order to work.

The major change from previous versions is that a connection needs to be authenticated by creating a Session before the CTI connection can be made. The Getting Started section details how the authentication change works with code examples to help.

The CTI related call control and event structure (using the PhoneManagerCTL class) has remained the same and should require very little, if any, changes to be made to the application.

4 Microsoft .Net Framework WCF

This is a local WCF service hosted on a Net.TCP named pipe connection that connects to the Phone Manager client running for the current *User*. This is not designed to replace the Phone Manager client but to enable simple commands to be sent and events to be received without having to implement the full Microsoft .Net Framework DLL but does require the Phone Manager client to be running for the User.

4.1 Getting Started

The application needs to establish a named pipe connection to the Phone Manager client. When Phone Manager starts it creates the named pipe connection on this extension.

```
net.pipe://localhost/CTI_{Username}
```

Where `{Username}` is the name of the user currently logged into the computer running Phone Manager.

The following example shows how to do this from a C# application.

The sample makes use of the MIcrosoft assembly `System.ServiceModel.ChannelFactory` (http://msdn.microsoft.com/en-us/library/system.servicemodel.channelfactory%28v=vs.100%29.aspx), so this needs adding as a reference to the project and adding in the using area.

References

```
using System.ServiceModel;
```

The WCF uses a `ServiceContract` so this need to be configured as an interface in the project. For each method that is going to be used the required interface `OperationContract` needs to be included. The interface can either be declared inside the application or referenced using the `Xarios.PhoneManager.ClientWCFService.ICTIInterface`. See the "Phone Manager API Reference.chm" for more details.

Call Control

```
[ServiceContract]
interface ICTIService
{
  [OperationContract]
  short MakeCall(string destination, string accountCode);
  [OperationContract]
  short AnswerCall(short callIndex);
  [OperationContract]
  short ClearCall(short callIndex);
  [OperationContract]
  short HoldCall(short callIndex);
  [OperationContract]
  short RetrieveCall(short callIndex);
  [OperationContract]
  short TransferCall(short callIndex, string destination);
  [OperationContract]
  short AnnouncedTransferCall(short callIndex, short heldcallindex);
}
```

To make a call, create a connection to the WCF service and call the `MakeCall` method.

Call Control Example

```
ChannelFactory<ICTIService> pipeFactory = new DuplexChannelFactory<ICTIService>(this,
new NetNamedPipeBinding(), new
extensionAddress(string.Format("net.pipe://localhost/CTI_{0}",
Environment.UserName)));
ICTIService cti = pipeFactory.CreateChannel();
short result = cti.MakeCall("08453736880", "");
pipeFactory.Close();
```

Telephony events can be subscribed to enable actions to be performed when the associated extension rings for example. To be able to receive the events the application needs be implement the `Xarios.PhoneManager.ClientWCFService.ICTIClient` interface and call the `RegisterForEvents` method.

Event Subscription

```
public class PhoneManagerExample : ICTIClient
{
   ChannelFactory<ICTIService> pipeFactory;
   ICTIService cti;

   public PhoneManagerExample()
   {
     pipeFactory = new DuplexChannelFactory<ICTIService>(this, new
NetNamedPipeBinding(), new
extensionAddress(string.Format("net.pipe://localhost/CTI_{0}",
Environment.UserName)));

     cti = pipeFactory.CreateChannel();
     if (cti == null) throw new Exception("Unable to connect to Phone Manager");

     cti.RegisterForEvents();
  }

  #region ICTIClient Members
  public void CallOffHook(CTICallDetails callInfo) {}
  public void CallRinging(CTICallDetails callInfo) {}
  public void CallAnswered(CTICallDetails callInfo) {}
  public void CallCleared(CTICallDetails callInfo) {}
  public void CallHeld(CTICallDetails callInfo) {}
  public void CallRetrieved(CTICallDetails callInfo) {}
  public void CallTransferred(CTICallDetails callInfo) {}
  public void AgentStateChanged(CTIAgentStateDetails agentStateInfo) {}
  public void Connected() {}
  public void Disconnected(){}
  #endregion

}
```

5 Command Line Executable

The Phone Manager executable enables some call control actions to be performed by calling this from a command line. Some 3rd party application support integration via the use of this method. The executable is in the installation folder where Phone Manager is installed and called PhoneManager.Client.exe.

For example, to make a call to 08453736880 from the command line: PhoneManager.Client.exe MC:08453736880.

See the Phone Manager User Guide for more details.

6 COM Component

This is a COM component that connects to the Phone Manager client running for the current *User*. This is not designed to replace the Phone Manager client but to enable simple commands to be sent and events received without having to implement the full Microsoft .Net Framework DLL but does require the Phone Manager client to be running for the User.

This is a wrapper around the Microsoft .Net Framework WCF API with both supporting the same set of commands and events to provide integration into non .Net environments.

6.1 Getting Started

The application needs to create an instance of the Phone Manager COM component. The class ID and prog ID values that can be used are shown below:

Prog ID: `PhoneManager.PhoneManagerCOM`

Class ID: `{A257A42C-CB00-411C-BB8F-51192DA00004}`

HTML & Javascript

```
<html>
    <head>
        <title>Phone Manager API Example - COM</title>
        <object id="phoneManager" name="phoneManager" classid="clsid:A257A42C-CB00-
411C-BB8F-51192DA00004"></object>
        <script language="javascript" type="text/javascript">
            function appendText(txt) {
                myTextBox.value = myTextBox.value + txt;
            }

            function phoneManager_onunload() {
                phoneManager.Dispose();
            }

            function butMakeCall_click() {
                appendText("Making call to " + txtNumber.value);
                appendText("\r\n");
                var returnCode = phoneManager.MakeCall(txtNumber.value);
                var msg = "myComComponent.HelloWorld returned " + returnCode;
            }
        </script>
    </head>
    <body onunload="phoneManager_onunload();">
        <h1>Phone Manager API Example - COM</h1>
        <table>
            <tr>
                <td>
                    <button id="butMakeCall" onclick="butMakeCall_click();">Make
Call</button>
                    <label>Number</label>
                    <textarea id="txtNumber" rows="1" cols="16"></textarea>
                </td>
            </tr>
            <tr>
```

```
                <td>
                    <textarea id="myTextBox" rows="10" cols="80"></textarea>
                </td>
            </tr>
        </table>

        <script for="phoneManager" event="CallOffHookEvent(callIndex, outsideNumber)"
language="javascript">
            function phoneManager::CallOffHookEvent(callIndex, outsideNumber) {
                appendText("CallOffHookEvent : " + callIndex + "\r\n" );
            }
        </script>

        <script for="phoneManager" event="CallRingingEvent(callIndex, outsideNumber)"
language="javascript">
            function phoneManager::CallRingingEvent(callIndex, outsideNumber) {
                appendText("CallRingingEvent : " + callIndex + "\r\n" );
            }
        </script>

        <script for="phoneManager" event="CallAnsweredEvent(callIndex,
outsideNumber)" language="javascript">
            function phoneManager::CallAnsweredEvent(callIndex, outsideNumber) {
                appendText("CallAnsweredEvent : " + callIndex + "\r\n" );
            }
        </script>

        <script for="phoneManager" event="CallClearedEvent(callIndex, outsideNumber)"
language="javascript">
            function phoneManager::CallClearedEvent(callIndex, outsideNumber) {
                appendText("CallClearedEvent : " + callIndex + "\r\n" );
            }
        </script>
    </body>
</html>
```

7 Macro Scripting (VBScript)

Phone Manager has a macro engine built into the client that enables VBScript macros to be written and triggered on call and other events. Call information can be used within the macros to provide information for integration.

For example this VBScript example will display a message box and display the caller id:

Example

```
MsgBox "Call from " & PhoneControl.CLI
```

7.1 Getting Started

The macros are built using the VBScript scripting language, for a detailed reference guide see (http://msdn.microsoft.com/en-us/library/d1wf56tt%28v=vs.84%29.aspx). From within the VBScript Phone Manager provides the *PhoneControl* object that allows access to the telephony specific actions and information.

8 TAPI (1st Party)

Phone Manager has a 1st party TAPI driver (TSP) available that supports both inbound and outbound calls. TAPI is an industry standard way of integrating a PBX into applications without having to perform the integration into each type of telephone system that is to be supported.

The TSP supports TAPI v2 and TAPI v3. 3rd Party TAPI is not supported.

Phone Manager provides all the features for screen popping and direct dialling out from applications using TAPI – but the 3rd party application must have implemented these features. If the application does not support these features then either they need to do this or another method of integration must be used.

See the Phone Manager User Guide for more details.

9 Contact Imports

The Communication Service that the Phone Manager clients connect to has the option to be able to import data and then make this data available with certain types of integration. Each line of data is associated with a telephone number and can contain up to 10 extra fields of information. For example the telephone number and a customer account number could be imported and then when a call is received with that telephone number the account number of the matching record could be used to screen pop their database record.

See the Phone Manager and Communication Service User Guide for more details.

10 Highlight and Dial

Phone Manager supports a highlight and dial feature. This works by highlighting a telephone number and then either double clicking on the Phone Manager banner icon or pressing a configurable hot key. As long as the number can be copied and pasted to and from the clipboard then this should work. This can be tested by highlighting the telephone number in the 3rd party application pressing Ctrl+C then opening notepad and pressing Ctrl+V, if the number then appears in notepad it should be ok.

See the Phone Manager User Manual for more details.

11 URI Schemes

Phone Manager supports a range of different well known URI schemes that enable applications to be able to do direct click to dial by turning the telephone numbers into hyperlinks. For example if an application wraps the telephone number into this format:

tel://01234567890

Clicking on this link will then cause this number to be dialled. The following URI formats are supported:

URI Scheme	*Example*
dial	dial://01234567890
sip	sip://01234567890
callto	callto://01234567890
tel	tel://01234567890
call	call://01234567890
dialfrompm	dialfrompm://01234567890