



OnCast Desktop for Cisco IP Phones Software Developer's Guide

OnCast Desktop Software Developers' Guide

Copyright © 2005 LiteScape Technologies, Inc. All rights reserved.

Information in this document is subject to change without notice. No part of this publication may be reproduced for any purpose other than the purchaser's personal use, without written permission of LiteScape Technologies, Inc.

LiteScape Technologies

1000 Bridge Parkway, Suite 200

Redwood City, CA 94065-1157

Voice 650.227.0220

Fax 650.227.0226

<http://www.litescape.com>

About LiteScape

LiteScape is a leading provider of software applications that enables the full potential of converged networks and emergent smart devices. LiteScape has developed powerful industry solutions on a multi-modal platform that delivers unsurpassed customer Point of Service solutions.

Trademarks

LiteScape, KIDS (Knowledge Information Delivery System), and OnCast are registered trademarks of LiteScape Technologies, Inc. in the U.S.A. and other countries. Microsoft and Windows are registered trademarks of Microsoft Corporation. All other trademarks and trade names referred to in this document are the property of other companies.

Revision History

Revision	Date Published	Reason for Revision	Comments
1.0	07/2005	New	

TABLE OF CONTENTS

About LiteScape	3
Trademarks	3
Revision History	3
OnCast Desktop Software Developer's Guide	5
Working with workflows	5
Anatomy of workflows	5
Sample payload file contents	6
Customizing Workflows	10
Editing workflow templates	10
Sample workflows file content	11
Code description	12
Building your own workflow plug-ins	12
Plug-in interface	13
Sample DLL	14
<i>Index</i>	18

OnCast Desktop Software Developer's Guide

OnCast Desktop is an IP broadcast message application that extends the functionality of Microsoft Outlook and lets you send broadcasts to IP telephones along with audio, video, Microsoft PowerPoint presentation and image files as well as surveys and RSS news articles.

You can also design workflows that let recipients reply to your messages with a wide range of options. This document describes workflows, how they work, and how you can extend their functionality. To use the information in this document effectively, you should understand software development, especially how to create Microsoft Windows DLL files.

Working with workflows

A workflow is one or more actions that you design for the recipient of your broadcast. Workflows consist of commands assigned to softkeys that are displayed on the recipients' IP phones when the message arrives. When the recipient presses a softkey assigned to an action, the action assigned to the key is generated. In some cases, you can create multiple sequential actions that serve as a workflow response from the recipient.

Five workflow templates are provided in OnCast Desktop, including Custom, Presentation, Survey, Stock and RSS. Each template is designed to be used with the type of message indicated by the template name. For example, the Presentation workflow template is designed to be used when you broadcast Presentation messages. Working with workflow templates and modifying them for your purposes is described in the section *Creating Workflows* in the **OnCast Desktop User Guide**.

You can also extend the power of workflows by customizing existing workflows and by building your own workflow plug-ins. The rest of this document addresses these procedures.

Anatomy of workflows

To understand the anatomy of a workflow, it helps to understand the anatomy of an OnCast Desktop message file. When a user saves a message to disk, the user specifies the file name, and OnCast Desktop adds the default extension .OCM (OnCast Message). These are ZIP files that can contain the following types of files depending on the content of the message.

- PNG image files for images and Microsoft PowerPoint presentation slides.
- WAV files for audio files.
- PAY for payload files, which contain message file information, author information, all message text, and the workflow settings. Every message file contains one payload file.

When a user selects a workflow template or designs a new workflow, OnCast Desktop saves the workflow settings in what's called a payload file as XML code. A sample is shown below.

Sample payload file contents

The following code provides the content of a payload file for a message that contains a recipient and subject along with text, one image, a three-slide Microsoft PowerPoint presentation, stock monitor entries, RSS news links and the Stock workflow template.

```
<?xml version="1.0"?>
<OnCastMessage>
  <Version>0.95</Version>
  <Header>
    <Type>PRESENTATION</Type>
    <Priority>NORMAL</Priority>
    <Sender>Charles Ackerman</Sender>
    <Recipients>
      <Recipient>/o=First Organization Litescape/ou=First Administrative
Group/cn=Recipients/cn=Charles.Ackerman</Recipient>
    </Recipients>
  </Header>
  <Message>
    <Prompt>Powered by LiteScape</Prompt>
    <Subject>here's your mixed message for the day</Subject>
    <Body>Please call me immediately. </Body>
  </Message>
  <Audio>
    <Type>NONE</Type>
    <Streaming>UNICAST</Streaming>
  </Audio>
  <Video>
    <FileName>
    </FileName>
  </Video>
  <Presentation>
    <Count>3</Count>
  </Presentation>
  <Survey>
    <Option>Bangkok Villa</Option>
    <Option>O Sole Mio</Option>
    <Option>Mario's La Fiesta</Option>
    <Option>O Chame</Option>
  </Survey>
  <Stocks>
    <Stock>
      <Symbol>MSFT</Symbol>
      <Rule>&lt;Rule&gt;&lt;LessThan&gt;50.00&lt;/LessThan&gt;&lt;GreaterThan&gt;100.00&lt;/GreaterThan&gt;&lt;/Rule&gt;</Rule>
    </Stock>
    <Stock>
      <Symbol>HPO</Symbol>
      <Rule>&lt;Rule&gt;&lt;LessThan&gt;20.00&lt;/LessThan&gt;&lt;GreaterThan&gt;40.00&lt;/GreaterThan&gt;&lt;/Rule&gt;</Rule>
    </Stock>
    <Stock>
      <Symbol>SUNW</Symbol>
      <Rule>&lt;Rule&gt;&lt;LessThan&gt;30.00&lt;/LessThan&gt;&lt;GreaterThan&gt;60.00&lt;/GreaterThan&gt;&lt;/Rule&gt;</Rule>
    </Stock>
  </Stocks>
```

```

<RSSFeeds>
  <RSS>
    <URL>http://news.yahoo.com</URL>

  <Rule>&lt;Rule&gt;&lt;Title&gt;IBM&lt;/Title&gt;&lt;Description&gt;server&lt;/D
escription&gt;&lt;/Rule&gt;</Rule>
  </RSS>
  <RSS>
    <URL>http://news.google.com</URL>

  <Rule>&lt;Rule&gt;&lt;Title&gt;IBM&lt;/Title&gt;&lt;Description&gt;server&lt;/D
escription&gt;&lt;/Rule&gt;</Rule>
  </RSS>
</RSSFeeds>
<Workflow>
  <Key>
    <Key>
      <Caption>Stocks</Caption>
      <Actions>
        <Action>
          <ID>Stock</ID>
          <Settings>
            <pluginValue>
              <settings>stock</settings>
            </pluginValue>
          </Settings>
        </Action>
      </Actions>
    </Key>
  </Key>
  <Key>
    <Key>
      <Caption>Chart</Caption>
      <Actions>
        <Action>
          <ID>Stock</ID>
          <Settings>
            <pluginValue>
              <settings>stockchart</settings>
            </pluginValue>
          </Settings>
        </Action>
      </Actions>
    </Key>
  </Key>
  <Key>
    <Key>
      <Caption>Headlines</Caption>
      <Actions>
        <Action>
          <ID>Stock</ID>
          <Settings>
            <pluginValue>
              <settings>StockHeadlines</settings>
            </pluginValue>
          </Settings>
        </Action>
      </Actions>
    </Key>
  </Key>
  <Key>
    <Key>
      <Caption>Browser</Caption>

```

```

    <Persistant>>true</Persistant>
    <Actions>
      <Action>
        <ID>Launch</ID>
        <Settings>
          <pluginValue>
            <settings>Launch Browser</settings>
            <URL />
          </pluginValue>
        </Settings>
      </Action>
    </Actions>
  </Key>
</Key>
</Workflow>
</OnCastMessage>

```

The contents of the file are defined below.

<pre><?xml version="1.0" ?></pre>	File header
<pre> <OnCastMessage> <Version>0.95</Version> <Header> <Type>RSS</Type> <Priority>NORMAL</Priority> <Sender>Charles Ackerman</Sender> <Recipients> <Recipient>/o=First Organization Litescape/ou=First Administrative Group/cn=Recipients/cn=Charles.Ackerman</Recipient> </Recipients> </Header> . . . </OnCastMessage> </pre>	OnCast Message header provides OnCast version number, tab active when the message was last saved, priority level, sender's name and recipients names.
<pre> <Message> <Prompt>Powered by LiteScape</Prompt> <Subject>here's your mixed message for the day</Subject> <Body>Please call me immediately. </Body> </Message> </pre>	Text message contents.
<pre> <Audio> <Type>FILE</Type> <Streaming>UNICAST</Streaming> </Audio> </pre>	Audio message contents and current cast setting - unicast or multicast
<pre> <Video> <FileName> </FileName> </Video> </pre>	Video message contents.
<pre> <Presentation> <Count>3</Count> </Presentation> </pre>	Presentation message settings with slide count.
<pre> <Survey> <Option>Bangkok Villa</Option> <Option>0 Sole Mio</Option> <Option>Mario's La Fiesta</Option> <Option>0 Chame</Option> </Survey> </pre>	Survey message options.
<pre> <Stocks> <Stock> <Symbol>MSFT</Symbol> </pre>	Survey message stock symbols and rule settings for upper and lower trigger levels.
<pre> <Rule>&lt;Rule&gt;&lt;LessThan&gt;50.00&lt;/LessThan&gt;&lt; GreaterThan&gt;100.00&lt;/GreaterThan&gt;&lt;/Rule&gt;</Rule> </Stock> </pre>	

```

<Stock>
  <Symbol>HPO</Symbol>

<Rule>&lt;Rule&gt;&lt;&lt;LessThan&gt;20.00&lt;/LessThan&gt;&lt;&lt;
GreaterThan&gt;40.00&lt;/GreaterThan&gt;&lt;/Rule&gt;</Rule>
</Stock>
<Stock>
  <Symbol>SUNW</Symbol>

<Rule>&lt;Rule&gt;&lt;&lt;LessThan&gt;30.00&lt;/LessThan&gt;&lt;&lt;
GreaterThan&gt;60.00&lt;/GreaterThan&gt;&lt;/Rule&gt;</Rule>
</Stock>
</Stocks>
<RSSFeeds>
  <RSS>
    <URL>http://news.yahoo.com</URL>
    <Rule>&lt;Rule&gt;&lt;&lt;Title&gt;IBM&lt;/Title&gt;&lt;&lt;
Description&gt;server&lt;/Description&gt;&lt;/Rule&gt;</Rule>
  </RSS>
  <RSS>
    <URL>http://news.google.com</URL>
    <Rule>&lt;Rule&gt;&lt;&lt;Title&gt;IBM&lt;/Title&gt;&lt;&lt;
Description&gt;server&lt;/Description&gt;&lt;/Rule&gt;</Rule>
  </RSS>
</RSSFeeds>
<Workflow>
  <Key>
    <Key>
      <Caption>Stocks</Caption>
      <Actions>
        <Action>
          <ID>Stock</ID>
          <Settings>
            <pluginValue>
              <settings>stock</settings>
            </pluginValue>
          </Settings>
        </Action>
      </Actions>
    </Key>
  </Key>
  <Key>
    <Key>
      <Caption>Chart</Caption>
      <Actions>
        <Action>
          <ID>Stock</ID>
          <Settings>
            <pluginValue>
              <settings>stockchart</settings>
            </pluginValue>
          </Settings>
        </Action>
      </Actions>
    </Key>
  </Key>
  <Key>
    <Key>
      <Caption>Headlines</Caption>
      <Actions>
        <Action>
          <ID>Stock</ID>
          <Settings>

```

RSS message news feed entries and article filter conditions.

Workflow buttons, in this case for Stock Monitor, which provides four buttons marked **Stocks**, **Chart**, **Headlines** and **Browser**. You can find a detailed description of workflow values in the section **Sample workflow file contents** below.

```
<pluginValue>
  <settings>StockHeadlines</settings>
</pluginValue>
</Settings>
</Action>
</Actions>
</Key>
</Key>
<Key>
  <Key>
    <Caption>Browser</Caption>
    <Persistant>true</Persistant>
    <Actions>
      <Action>
        <ID>Launch</ID>
        <Settings>
          <pluginValue>
            <settings>Launch Browser</settings>
            <URL />
          </pluginValue>
        </Settings>
      </Action>
    </Actions>
  </Key>
</Key>
</Workflow>
```

When the message is sent, the contents of the payload file are copied into memory and used to process the message.

Customizing Workflows

Software developers can customize workflows two ways:

- Edit the file `workflows.xml` located in `Program Files\LiteScape\oncast`. For more information, see the section **Editing workflow templates** below.
- Create your own workflows by designing DLL plug-ins. For more information, see the section **Building your own workflows** below.

Editing workflow templates

You can create and modify existing workflow templates by editing the file that contains all workflow information called `workflows.xml`. You can find this file located in `Program Files\LiteScape\oncast`. This is a text file containing standard xml tags. You can use the Microsoft Windows Notepad to edit the file or your own favorite text editor or xml editor as long as they save simple text files.

You can save your current workflow by clicking **File > Save > Workflow**, select the location where you want to save the file, then specify a file name.

You can view a sample of `workflows.xml` contents below. A short description of the xml code follows the sample.

Sample workflows file content

A section of the contents of `workflows.xml` is shown below.

```
<?xml version="1.0"?>
<Workflows>
  <Workflow Name="[Presentation]">
    <softkey>
      <name>Prev Slide</name>
      <actions>
        <action>
          <plugin>Prev Slide</plugin>
          <pluginValue></pluginValue>
        </action>
      </actions>
    </softkey>
    <softkey>
      <name>Next Slide</name>
      <actions>
        <action>
          <plugin>Next Slide</plugin>
          <pluginValue></pluginValue>
        </action>
      </actions>
    </softkey>
  </Workflow>

  <Workflow Name="[Survey]">
    <softkey>
      <name>Select</name>
      <actions>
        <action>
          <plugin>Survey</plugin>
          <pluginValue><Survey /></pluginValue>
        </action>
      </actions>
    </softkey>
  </Workflow>
```

Two buttons are defined in the sample above - **Prev Slide** and **Select**. The first button is part of the Presentation workflow, while the second button is part of the Survey workflow. You can see this in the code that surrounds each key definition.

Code description

- There are four possible elements - **workflow name**, **softkey name**, **action name** and **action plug-in**.
- Each workflow button is bounded by

```
<Workflow Name="[name]"><softkey> . . . </softkey></Workflow>
```
- Each button name is bounded by

```
<name>button name</name>
```
- Each set of actions is bounded by

```
<action><actions> . . . </action></actions>
```
- An action is defined with a plug-in name and plug-in value this way -

```
<plugin>name</plugin>
<pluginValue><value /></pluginValue>
```
- There are five plug-in names and values - **Custom**, **Presentation**, **Survey**, **Stock Monitor** and **RSS**.

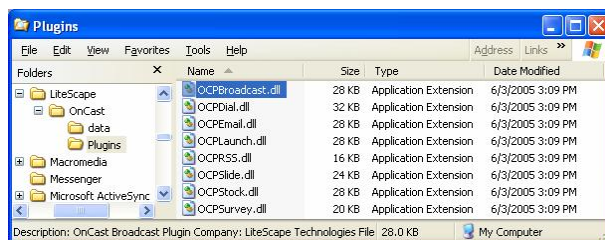
Building your own workflow plug-ins

You can build your own Microsoft Windows DLL files that can serve as plug-ins for your work. For example, you might want to create a custom plug-in called the SMS plug-in that sends an SMS message to people based on responses they give when they click a softkey. You can design the SMS plug-in to launch an application on the desktop and pass an SMS phone number you define in the Instant Workflow window. The results is similar to the Email plug-in, however, the SMS plug-in sends an SMS, not an email message.

OnCast Desktop provides four plug-ins for all existing workflows :

- OCPDial.dll dials phone numbers.
- OCPBrowse.dll launches Internet Explorer. This passes the path and IE executable name to Microsoft Windows.
- OCPEmail.dll sends email messages.
- OCPLaunch.dll launches a specified application. This passes the path and application executable name to Microsoft Windows.

You can find these files in the folder Plugins located directly under your OnCast program file folder, most likely C:\Program Files\LiteScape\OnCast\Plugins, as shown below.



The other four DLL files listed above provide the functionality for the default workflow templates described earlier.

Plug-in interface

You can create your own plug-ins as long as they subscribe to the interface defined below. A sample DLL is provided.

```

Begin -----

using System;
using System.Collections;

namespace OnCast
{
    public interface IPlugin
    {
        IPluginHost Host {get;set;}

        string Id {get;set;} // Plugin ID
        string Name {get;} // Plugin Name
        string Description {get;} // Plugin Description
        string Author {get;} // Plugin Author
        string Version {get;} // Plugin Version

        int pluginType(); // Type of Plugin, 1: Workflow (we don't have any
            other type for now!)
        System.Windows.Forms.UserControl ShowPreferences {get;} // Settings
            window
        string GetDefaultSettings(); // Default settings for this plugin
        void updateSettings(string xmlData); // to set plugin's setting
        void Execute(Hashtable args,string xmlData); //
        bool hasPreferences(); // to check if plugin has any preferences or not
        bool AllowWorkflow(); // The check if we can show it in the workflow
            list or not (we can have "hidden" plugins as well)
        void About(); // plugin's about
        void Initialize(); // To initialize
        void Dispose(); // To dispose
    }

    // These are the functions that Plugin Host (OnCast) gives to plugins

    public interface IPluginHost
    {

        void sendMail(string subject, string body); // Send Email
        void processTemplate(string xmlPayloadFile,bool RespondOnly,string
            ipDevice, string userName); // Broadcast
        string fetchPluginData(string xmlPayloadFile,string command, string
            pluginName); // retrieve plugin's data
        void dialNumber(string phoneNumber, string passCode, string ipDevice,
            string userName); // Dial
        void launchBrowser(string username, string url); // Launch Browser
        void launchApplication(string username, string appName); // Run
            Application
        Payload LoadPayload(string PayloadFile); // Load an OCM file and access
            the content through "Payload" structure
    }
}

```

End -----

Sample DLL

The code for a sample DLL is provided below.

```
using System;
using System.IO;
using System.Xml;
using System.Collections;
using OnCast;

namespace OCPBroadcast
{
    /// <summary>
    /// Simple Plugin for OnCast Desktop to broadcast another OCM file.
    /// </summary>
    ///
    public class OCPBroadcastPlugin : IPlugin
    {
        public OCPBroadcastPlugin()
        {
            //
            // Nothing to do here!
            //
        }

        //Declarations of all our internal plugin variables
        string myId = "OCPBroadcast";
        string xmlData = String.Empty;
        string myName = "Broadcast";
        string myDescription = "OnCast Instant Workflow Plugin to
Broadcast A Form.";
        string myAuthor = System.Windows.Forms.Application.CompanyName;
        string myVersion =
System.Windows.Forms.Application.ProductVersion;
        IPluginHost myHost = null;
        System.Windows.Forms.UserControl myMainInterface = new
OCPBroadcastPluginInterface();

        public int pluginType()
        {
            return 1; //workflow
        }
        /// <summary>
        /// Description of the Plugin's purpose
        /// </summary>
        public string Description
        {
            get {return myDescription;}
        }

        /// <summary>
        /// Author of the plugin
        /// </summary>
        public string Author
        {
            get {return myAuthor;}
        }
    }
}
```

```
/// <summary>
/// Host of the plugin.
/// </summary>
public IPluginHost Host
{
    get { return myHost; }
    set { myHost = value; }
}

public string Name
{
    get { return myName; }
}

public bool AllowWorkflow()
{
    return true;
}

public string ShowSettings(string xmlData)
{
    return null;
}

private string FormatXML(XmlDocument doc)
{
    // Create a stream buffer that can be read as a string
    StringWriter sw = new StringWriter();

    // Create a specialized writer for XML code
    XmlTextWriter xtw = new XmlTextWriter(sw);

    // Set the writer to use indented (hierarchical) elements
    xtw.Formatting = System.Xml.Formatting.Indented;

    // Write the XML document to the stream
    doc.WriteTo(xtw);

    // Return the stream as a string
    return sw.ToString();
}

public string GetDefaultSettings()
{
    System.Type myInterface = myMainInterface.GetType();
    string xmlData = (string)
myInterface.InvokeMember("outputData", System.Reflection.BindingFlags.InvokeMeth
od, null, myMainInterface, null);
    return xmlData;
}

public void updateSettings(string xmlData)
{
    System.Type myInterface = myMainInterface.GetType();
    object[] args = {xmlData};
    object result =
myInterface.InvokeMember("processData", System.Reflection.BindingFlags.InvokeMet
hod, null, myMainInterface, args);
}

public bool hasPreferences()
{
    return true;
}
```

```

    }

    public void Execute(Hashtable args, string xmlData)
    {
        string command = (string) args["command"];
        string pluginData =
this.Host.fetchPluginData(xmlData,command,myName);
        ArrayList dataForTemplates =
parseXMLForTypeofAnswer(pluginData);
        if (dataForTemplates.Count > 0)
        {
            string respondOnly = (string) dataForTemplates[1];
            string ipDevice = (string) args["ipDevice"];
            string username = (string) args["username"];
            if (respondOnly.Equals("true"))
            {
                this.Host.processTemplate((string)
dataForTemplates[0],true,ipDevice,username);
            }
            else
            {
                this.Host.processTemplate((string)
dataForTemplates[0],false,ipDevice,username);
            }
        }
    }

    public ArrayList parseXMLForTypeofAnswer(string xmlData)
    {
        bool success = false;
        string tagName = String.Empty;
        ArrayList outputData = new ArrayList();
        //XmlTextReader reader = new XmlTextReader(xmlData);
        XmlTextReader reader = new
XmlTextReader(xmlData, System.Xml.XmlNodeType.Document, null);
        try
        {
            while ((reader.Read()) && (! success))
            {
                switch (reader.NodeType)
                {
                    case XmlNodeType.Element:
                    {
                        tagName = reader.Name;
                        if
(tagName.Equals("pluginValue"))
                        {
                            reader.ReadStartElement("pluginValue");
                            outputData.Add(reader.ReadElementString("template"));
                            outputData.Add(reader.ReadElementString("respond"));
                            reader.ReadEndElement();
                            success = true;
                        }
                        break;
                    }
                }
            }
        }
        catch (Exception ex)
        {

```

```
        Console.WriteLine("Found an error in processData for
FeedBackSInterface class: " + ex.ToString());
    }
    return outputData;
}

public void About()
{
    // if you want to show the about box...
}

public System.Windows.Forms.UserControl ShowPreferences
{
    get { return myMainInterface; }
}

public string Version
{
    get { return myVersion; }
}

public string Id
{
    get { return myId; }
    set { myId = value; }
}

public void Initialize()
{
    //This is the first Function called by the host...
    //Put anything needed to start with here first
}

public void Dispose()
{
    //Put any cleanup code in here for when the program is
stopped
}
}
```

Index

- A**
 - anatomy of workflows, 5
- B**
 - building plug-ins, 12
- C**
 - customizing workflows, 10
- D**
 - description of workflow code, 12
 - dll files listed, 12
- E**
 - editing workflow templates, 10
- F**
 - files, payload, 5
- I**
 - interface, plug-in, 13
- O**
 - OCPBrowse.dll, 12
 - OCPDial.dll, 12
 - OCPEmail.dll, 12
 - OCPLaunch.dll, 12
- P**
 - payload files, 5
 - plug-in interface, 13
 - plug-ins, building, 12
- S**
 - sample workflows, 11
- T**
 - templates for workflows, editing, 10
- W**
 - workflow anatomy, 5
 - workflow code description, 12
 - workflow templates, editing, 10
 - workflows, customizing, 10
 - workflows, working with, 5
 - working with workflows, 5