Moderro
IEM Management Software

Developer Guide

Release 2.6

Moderro Technologies

www.moderro.com

Table of Contents

Preface

This preface describes the purpose, audience, content organization, and conventions in this guide, and provides information on the related documents.

- Purpose
- Audience
- Organization
- Document Conventions
- Related Documentation

Purpose

This developer guide provides information about the Moderro Interactive Experience platform system. It describes different APIs for the IEM server and configuration for widgets. Additionally, it has information about standard and non-standard global objects.

Audience

This guide is intended for web developers and application programmers who will develop applications for the kiosk.

Organization

Title	Purpose
System Overview	Provides an overview of the Moderro Interactive Experience Platform system.
APIs for the IEM Server	Provides information about APIs for the IEM server, including how to use a service API and service API documentation.
Widgets	Provides an overview of the supported widgets that can be configured and controlled using javascript.
JavaScript Global Objects	Provides an overview on the javascript global objects that allow web applications to be better integrated with the IEC.

Document Conventions

Convention	Indication
bold font	Commands, keywords, and user-entered text appear in the **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you assign values are in the *italic* font.
[ ]	Elements in the square brackets are optional.

{x | y | z}	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A non-quoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
courier font	Terminal sessions and information that the system displays appear in the courier font.
courier bold font	Command names and samples appear in the **courier bold** font.
< >	Non-printing characters, such as passwords, are in the angle brackets.
[ ]	Default responses to the system prompts are in the square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.
Option > Option	Used to describe a series of menu options.

Note Means *reader take note*. Notes contain suggestions or references to materials that are not covered in the guide.

Related Documentation

Moderro Interactive Experience Platform Installation Guide

System Overview

Last Revised: October 27, 2016

Chapter Overview

This guide is intended for web developers and application programmers who will develop applications for the kiosk.

This chapter provides an overview of the Moderro Interactive Experience Platform

system. The topics in this chapter include:

- System Overview
  - Properties
  - Management System
  - Management Protocol
  - Applications
  - Thin Client Features
  - Security
  - Application Development
  - JavaScript Injection
  - User-Agent Rules
  - Content Caching
  - Off-Line Caching
- Cobra Browser
- Cobra JavaScript API
- Japanese Fonts

System Overview

This section contains some details to help you to understand how the entire platform works.

Moderro's Interactive Experience Platform includes two components: Moderro Interactive Experience Client (IEC), which is the thin client, and Moderro Interactive Experience Manager (IEM), which is the management server. The IECs can be configured and controlled remotely using the IEM.

The IEC is a Linux-based device, which is designed to run feature rich HTML and JavaScript applications. The central part of its software is a specially designed embedded web browser called Cobra.

Properties

IEC software is configured and controlled using properties. A property is similar to a Microsoft Windows Registry item, but provides much more functionality. For example, in contrast to a registry item, the IEC properties have a very rich type system and have an ability to subscribe to property changes.

A software engineer will declare a property with some type. In contrast to other systems, the IEC supports virtually any type - an integral type (e.g. enumerate, integer, boolean, real, etc.), a collection type (e.g. structure, list, map, hash, set, etc.), or even a composition of those types (e.g. list of structures or a field of the structure can be a map of an integer or a vector of a string).

A property can be declared persistent, which means that it can contain a configuration parameter for the application. At the same time that property can be used from another application to control or get information from the application, which owns the property. That property can be replicated to and from the IEM, and thus, provide a way to configure the application remotely.

Management System

The IEM is an appliance. Its user interface is implemented as a web application. It does not require the installation of any software on the administrator's computer. The user interface's terminology is similar to those of Microsoft Management Server. For example, the IEM uses the terms that a Windows administrator would be familiar with such as domain, group, profile, and policy.

The IEM requires HTTPS to access it from the IECs that you want to control and from the administrators' computers.

In contrast to other management systems, the IEM does not require the IEC and the management system to be within the same network. Hence, the IECs can be located behind firewalls. To support firewalls, the IEM's management protocol is designed as a one way protocol, which means the IEC always initiates the communication.

The management protocol is based on HTTPS, consists of requests and responses, and is synchronous. In addition to HTTPS, the IEM protocol uses a token-based device authentication algorithm to increase security. When an IEC is registered in the IEM, it receives a token from the IEM. In every request the IEC must provide that token to be authenticated in the IEM. In every response, the IEM sends a new token, which must be used in the next IEC request.

The IEM supports different versions of IEC firmware at the same time.

Existing systems can be integrated with the IEM using a protocol based on XML and HTTPS.

Management Protocol

The following is a list of requests that the current revision of management protocol supports:

- Is available: Asks the IEM if it is available. The IEM can reply that it is not available such as when it is undergoing maintenance.
- Is registered: Asks if the IEC is registered in the IEM.
- Register: Registers the IEC in the IEM. To perform this request successfully, the device must provide a name and a password of a user who is registered in the management system and has the right to register devices.
- Unregister: Unregisters the IEC in the IEM. To perform this request successfully, the device must provide a name and a password of a user who is registered in the IEM and has the right to unregister devices.

- Load device information: Loads IEC information from the IEM including device name, description, and location.
- Save device information: Saves IEC information to the IEM.
- Load device profile: Loads the IECs properties from the IEM.
- Save device profile: Saves the IEC's properties to the IEM.
- Sync device profile: Synchronizes the IEC's properties with the IEM.
- Ping: Reports IEC's device status to the IEM.
- Save events: Saves IEC's device events to the IEM.
- Save screenshot: Saves a screenshot from the IEC to the IEM.

Applications

The core application in this solution is a web browser named Cobra. Thus its applications are essentially web applications.

Cobra is not only a browser, but also an application environment. That means it contains many features to meet specific needs of customer applications. For example, it allows you to control the device entirely. From a web application you can access any property of any application in the system, so, you have a full control over the device your application is running on.

In addition to that, you can easily control peripherals connected to the system such as printers, web cameras, and other devices connected via the serial port. For example, you can print a resource given by a URL (i.e. HTML page, PDF document, or image) from your web application without a print dialog box appearing or set all the printing parameters for your printing job (i.e. paper size, resolution, etc.). All those features are implemented asynchronously so you do not stop your web application animations or interactivity.

There are also many specialized widgets available that can be used in your web applications. The widgets provide a functionality which is very difficult to implement in JavaScript. For example, the VideoPlayer widget supports most video and audio formats and codecs. In contrast to regular video players implemented as NP plugins, the VideoPlayer widget has an easy to use JavaScript API and features that other players lack such as smart caching. Another example is the Ticker widget. Although the Ticker widget can be implemented in JavaScript, it works much faster being implemented in C++. See the *Moderro Interactive Experience Platform Content Creation Guidelines* for the widgets.

Thin Client Features

The thin client, otherwise known as the IEC, contains many features that you should be aware of in order to build applications for it:

- Security
- Application development
- Third party systems integration
- JavaScript injection
- User-agent rules
- Content caching
- Off-line caching

Security

- Virus free environment: Since the IEC is a Linux-based device, it cannot be infected by viruses.
- Read only file system: All file systems that contain device software are read-only file systems. They cannot be remounted in read-write mode. Thus you can be sure the device software is not altered by an intruder. The IEC has two read-write file systems: a small read-write file system to store settings and another to store content cache. Binary applications, however, cannot be run from those file systems.
- No third party binary software: The IEC does not allow any third party binary software to run on it. The only type of applications allowed are web applications (i.e. applications written using HTML and JavaScript). This type is totally controlled as they are executing in the browser's security sandbox.

Application Development

- APIs: The IEC can only run web applications. However, in contrast to regular desktop web browsers, the IEC provides proprietary APIs to create feature rich applications. Those APIs allow applications to communicate with various types of peripherals, such as cameras, printers, optical scanners, barcode scanners, magnetic card readers, remote controls, etc. It is even possible to control a peripheral via a serial port from your web application.
- Feature rich widgets: In addition to APIs, the IEC provides visual components to enhance the capabilities of your application. The PDF viewer and video player are examples of widgets. Instead of viewing PDF documents or playing video in desktop web applications as you regularly do, you will be able to have complete control over those parts of your application. For example, you can open a PDF document on any page, turn over the pages, print any page, etc. You cannot do that with a regular PDF reader plug-in within your desktop browser.
- Touch screens support: The IEC supports a wide variety of touch screens.
- Dual head support: The UEC can work with two monitors at the same time.
- Voice over IP support: The IEC supports IP telephony using Session Initiation Protocol (SIP). It is compatible with Cisco VoIP products, such as Cisco Unified Communications Manager (CUCM).
- Video conference support: Applications can easily utilize its video conferencing capabilities.

JavaScript Injection

This feature allows you to incorporate JavaScript code into an alien web page. This feature gives a way to modify existing web content to better fit in the current application environment. If you want to show a web site as a part of your application in iframe and that alien web site contains controls to do a search, you can remove those controls with JavaScript injection.

To turn this feature on you must set two properties:

1. The browser.content.javascript.injection.enabled property must be set to ‘true’.
2. The browser.content.javascript.injection.rules must contain injection rules. This property is a map with URL wildcard as a key and JavaScript code as a value.

Both properties are runtime properties so you are able to change their values inside your application and those changes become effective immediately.

With this feature turned on, Cobra loads an URL and checks that URL against injection rules. If one rule matches, Cobra executes the corresponding JavaScript code from that rule for the current window object. If multiple rules match, the code from all those rules will be executed.

Note Cobra's global JavaScript objects are registered before any injection so you are able to use them in the injected code.

The following example shows how to use JavaScript injection in an application (file.../.../test/js/injection.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

    <head>5
    <title>..:: JavaScript injection example ::.. </title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <style>

        body
        {
        margin: 20px;
        background-color: #000000;
        color : #eeeeee;
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
        color : #eeeeee;
        }

    </style>

    <script type="text/javascript">

        function init ()


        {
        global . registry .setValue("browser.content. javascript . injection .enabled", "true");
        global . registry .setValue("browser.content. javascript . injection . rules",
        '<map><item><key>*osnews*</key><value>function hide () { document.getElementById("sidebar").style.display =
        "none"; } addEventListener("load",  _hide_, false);</value></item></map>');

        document.getElementById("content").src = "http://www.osnews.com";
        }

    </script>

    </head>

    <body onload="init()">

    This application hides sidebar on OS News web site.
    <br />
    <br />

    <iframe id="content" width="1024" height="768" />

    </body>

</html>
```

Sample code 2 (file ../../test/js/useragent-print.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
  <html>
```

```
<head>

    <title>..:: Print User-Agent ::..</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<style>

    body
    {
    margin: 20px;
    background-color: #ff0000;
    color: #eeeeee;
    font-weight: bold;
    font-family: Arial;

    font-size: 20px;
   }


</style>

<script language="JavaScript">

    function init ()
  {
    document.getElementById("userAgent").innerHTML = '"' + navigator.userAgent + '"';
  }

</script>

</head>

<body onload="init()">

    Browser User-Agent is: <span id="userAgent" />

</body>

</html>
```

Content Caching

When you develop an application that must be usable when you have limited bandwidth, content caching becomes a very important feature of your application.

Regular desktop browsers do not give your application the ability to fully control content caching but Cobra has special APIs to control content caching. Regular web content like HTML documents, JavaScript code, CSS code, images, etc. are distinguished from media content like video and audio files. There are two different APIs to control caching for both types of content: global.networkCache and global.mediaCache.

Off-Line Caching

The user can set properties within a device profile or applied policy in the IEM that enables aggressive caching on an IEC. As a result, web and media content is cached by the IEC so that if the IEC becomes off-line (connection to the startup URL is lost), it can still display the content that users had previously interacted with before going off-line.

Note Only static page content is cached. Images and embedded videos are also cached. Dynamic page content is NOT cached.

In order to activate aggressive caching, you must first configure property settings in the IEC's device profile or an applied policy within the IEM. Refer to the *Moderro Interactive Experience Client 4600 Series User Guide* for instructions on how to configure off-line caching.

Cobra Browser

The name Cobra is derived from COBALT Browser, where COBALT is a C++ library that is used for embedded applications device software.

As explained above, Cobra is both a web browser and application environment. It contains features for kiosk specific web applications. In contrast to regular desktop browsers, such as Mozilla Firefox or Google Chrome, Cobra is not only a browser for HTML documents, but a platform for feature rich applications. This platform is tightly integrated with device's hardware, its capabilities are far beyond the capabilities of regular desktop browsers. For example, in your HTML and JavaScript application you can easily implement a video conferencing or a communication with monitor, connected to the device via serial port.

From a web application developer's prospective, those new features are represented by new JavaScript objects and functions introduced by Cobra.

Cobra assumes that a touch screen is connected and end users are operating it with either fingers or a stylus. A mouse and keyboard are supported also.

There are specialized widgets to use in kiosk web applications. Those widgets provide a functionality which is very hard if even possible to implement in JavaScript. As an example, there is a videoplayer, which supports most of video and audio formats and codecs. In contrast to regular video players implemented as NP plugins, this widget has a very rich and easy to use JavaScript API, has some interesting features (like smart caching) other players lack. Another example is the ticker widget, which can be implemented in JavaScript, but being implemented in C++ works much faster. In addition to regular plugins support (like Adobe Flash Player), Cobra provides several proprietary widgets to simplify developer's life. Those widgets can be configured and controlled from JavaScript.

Cobra JavaScript API

In contrast to statically typed programming languages like C and C++, there is no common way to declare JavaScript classes, object prototypes, and functions. However, it is very helpful for a developer to have such a description as a reference. So, instead IDL-like notation is used to solve this problem. The following is an example of IDL-like notation:

```
interface Factorial
  {
  readonly attribute int lastValue; int
  value(in int n) const;
  };
```

In the example above, it declares a Factorial interface which allows you to calculate n! with function value() and to get the last calculated value using lastValue read-only attribute.

Note Consider the statement "object A implements interface B". However, if you try to call typeof() function for object A you will not see B as the result. The result will be Object. Interfaces are introduced here to simplify objects description.

Cobra introduces a new communication mechanism for JavaScript objects. It is called signals and slots, a mechanism which comes from NOKIA's Qt library that Cobra had been built upon. Signals

and slots are used for communication between objects. The signals and slots mechanism is a central feature of Cobra JavaScript API. In GUI programming, when one widget is changed, we often want another widget to be notified. More generally, we want objects of any kind to be able to communicate with one another.

For example, if a user clicks a Close button, the desire is to have the window's close() function to be called. A signal is emitted when a particular event occurs. Cobra's JavaScript classes have many predefined signals. As an example of signal-slot mechanism usage network operations can be mentioned.

Since web applications are primarily related to network, the majority of new JavaScript classes, introduced by Cobra, are network-related as well. So, to allow web application to have a responsive user interface, all those new classes perform asynchronously. To support that asynchrony, Cobra uses signals and slots.

Note At first, you can think about signals like about JavaScript events with a different API, and about slots like about JavaScript event handlers.

Example

The following is a Clock JavaScript class interface:

```
interface Clock
{
readonly attribute Date timestamp; signals:

void tick();
};
```

This interface declared signal tick() which is fired by the object every second. To use this signal in JavaScript you can write the following code.

```
var clock;

...

function updateClock
{
document.getElementById("textClock").innerHTML = clock.timestamp.toString();
}

function init()
{
clock.tick.connect(updateClock);
}
```

Assuming function init() is called when HTML document is loaded, that function connects signal tick() to slot updateClock(). As the result the innerHTML attribute of DOM element with textClock id will be changed to current timestamp every second.

Japanese Fonts

For 2.5, four new Japanese fonts were added:

1. IPA Gothic
2. IPA P Gothic
3. IPA Mincho

4. IPA PMincho

To use the fonts in applications, use the following values for the font-family variable:

font-family: "IPAGothic";

font-family: "IPAPGothic";

font-family: "IPAMincho";

font-family: "IPAPMincho";

APIs for the IEM Server

Chapter Overview

This chapter provides information about APIs for the IEM server. Topics in this document include:

- "How to Use a Service API"
  - "API Authorization"
  - "Input Data Format"
  - "Output Data Format"
  - "General Notes"
  - "Performance Service Examples"
  - "Basic API for Device List, Create, Edit, List of Groups, and Profile Properties (examples)"
  - "Schedule Rules Format and Examples"
- "Service API Documentation"
  - "Account Service"
  - "Address Service"
  - "Authentication Service"
  - "Company Service"
  - "Contact Service"
  - "Country Service"
  - "Device Service"
  - "Device_model Service"
  - "Event Service"
  - "Firmware Service"
  - "Group Service"
  - "Image Service"
  - "Notification Service"
  - "Person Service"
  - "Policy Service"
  - "Product Service"
  - "Profile Service"
  - "Role Service"
  - "Schedule Service"
  - "Specification Service"
  - "System Service"
  - "User Service"

How to Use a Service API

API Authorization

When calling the service, append the parameter “auth” at the end:

&auth={“account”:”account_name”,”user”:”user_login”,”password”:”user_password”}

For example:

http://[service host]//xml/?service=specification.set_action_property&args=[69,”screenmonitor.update”, 1]&auth={“account”:”Root”,”user”:”Administrator”,”password”:”TopSecret”}

Input Data Format

The XML gate URL is **http://your-service-host/xml?[service].[method]** where:

[service] - service name (auth, account, device,...)

[method] - service method name

You can use either the XML (recommended for long data input) or JSON (recommended for small data input) arguments format.

If you want to use the XML input, you must send a GET or POST request with one argument "data". The argument "data" must contain XML with input parameters in this format:

Use the GET or POST parameter "data" to send arguments in XML format:

```
<xml>
    <param1_name>[param1_value]</param1_name>
    <param2_name>[param2_value]</param2_name>
    <param3_name>[param3_value]</param3_name>
    ...
</xml>
```

where:

param[X]_name - name of argument X

param[X]_value - value of argument

X

For example:

http://your-service-host/xml/?service=user.get_by_id&data=<xml><id>1</id></xml>

data is:

```
<xml>
    <id>1</id>
</xml>
```

First argument is called "id" and has the value 1.

If you want to use the JSON input, you must send GET with the argument “args”. The argument "args" must contain a JSON-object or JSON-array with arguments.

For example:

http://your-service-host/xml/?service=user.get_by_id&args={"id":"1"}

OR

http://your-service-host/xml/?service=user.get_by_id&args=["1"]

Sometimes you need to pass arrays. In the JSON-format, they are transmitted in the form of json-arrays. In XML, elements of the array pass in item-tags. For example:

http://your-service-host/xml/?service=xxx.xxx&data=<xml><id><item index="1">first</item><item index="2">second</item></id></xml>

OR

http://your-service-host/xml/?service=xxx.xxx&args={"id":["first","second"]}

OR

http://your-service-host/xml/?service=xxx.xxx&args=[["first","second"]]

To use the methods, you need to be logged on. However, you can perform individual searches without having to log in. To do this, you need to transfer data to authorize an additional JSON-object:

auth={"user":"MyUser","account":"MyAccount","password":"MyPass"}

Example:

http://your-service-host/xml/?service=user.get_by_id&args=["1"]&auth={"user":"MyUser", "account":"MyAccount","password":"MyPass"}

If its parameter is set, the request is processed with "auth" authentication data. The session (normal) authentication will be ignored.

Output Data Format

The answer comes in this XML format:

```
<xml>
    <status>
        <code>[ OK | FAILURE | ERROR]</code>
        <message>[Human-readable result message string]</message>
    </status>
    <data>
        [Additional data]
    </data>
</xml>
```

where CODE is result-type code:

- OK - request is successful
- FAILURE - request failed but it is not an error (i.e., the login was incorrect)
- ERROR - error occurred during request

Note The data tag may be empty.

General Notes

Many methods, especially create, update, and delete, take the input parameter object. This parameter is analogous to the output from the get_by_id method. It has to be provided in JSON format. The format of the object and the required fields can be viewed as a result of the output the method object (), which almost every service has. This method is needed just for this purpose.

Parameter names are validated. They must start with an English letter and contain at least three characters which could be letters, numbers, dashes, or underscores. Otherwise, there will be a validation error.

Performance Service Examples

http://[service]//xml/?service=event.index&args=["device","_deviceID_","_limit_","_(severity separated by comma, empty means all)_","_(facility separated by comma, empty means all)_"]

For example:

http://[service]//xml/?service=event.index&args=["device","168","100","ERROR,INFO","replicator,co"]

Performance-Event:

http://[service]//xml/?service=event.device_perfomance_info&args=["_device_ID_","_limit_"]

For example:

http://[service]/xml/?service=event.device_perfomance_info&args=["168",

"100"]

The list of possible facility for a device:

http://[service]/xml/?service=event.get_facility_list&args=["device","_deviceID_"]

For example:

http://[service]//xml/?service=event.get_facility_list&args=["device","168"]

The severities are standard: ERROR, INFO, WARNING, etc.

When sending notifications to a URL, the data comes in the following format:

```
< ? Xml version="1.0" encoding="UTF-8"?>
<xml>
    <event>
        <severity>INFO</severity>
        <module>system</module>
        <message>(test: INFO / system) USB device</message><data>
        <item type="string" name="usb-device"><![CDATA[/dev/sdb1]]></ item>
        <item type="string" name="usb-model"><![CDATA[DISK 2.0]]></item>
        <item type="string" name="usb-product"><![CDATA[EMERGENCY]]></ item>
        <item type="string" name="usb-vendor"><![CDATA[USB]]></ item>
        </data>
    </event>
    <event>
        <severity>ERROR</severity>
        <module>replicator </module>
        <message>(test: ERROR / replicator) Cannot open file \ '/ persistent / MODEL \' (No such file or
        directory).</message>
        <data></data>
    </event>
    .....
```

</xml>

Basic API for Device List, Create, Edit, List of Groups, and Profile Properties (examples)

http://[service]//xml/?service=device.get_by_serial&args=["300074500007"] Or by ID:

http://[service]//xml/?service=device.get_by_id&args=[

2465] Create:

https://[service]//xml/?service=device.create&args=[{"account_id":"ACCOUNT_ID","serial_number":"SERIAL_NUMBER", ... The rest of fields are optional... }] &auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

Edit:

Note Only send the fields that you need to change.

https://[service]//xml/?service=device.update&args=[{"id":"DEVICE_ID", ... The rest of fields are optional ... }] &auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

List of groups:

https://[service]/xml/?service=group.get_tree_by_objects&args={"TYPE","ACCOUNT_ID","AS_TREE"}&auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

TYPE - device or user

ACCOUNT_ID - account id

AS_TREE - 1 (output as tree), 0 (output as

list) List of profile properties of user or device:

- Device tree:

  https://[service]/xml/?service=profile.device_tree&args={"device_id":"DEVICE_ID"}&auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

- Device list:

  https://[service]/xml/?service=profile.device_list&args={"device_id":"DEVICE_ID"}&auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

- User tree:

  https://[service]/xml/?service=profile.user_tree&args={"user_id":"USER_ID"}&auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

- User list:

  https://[service]/xml/?service=profile.user_list&args={"device_id":"DEVICE_ID"}&auth={"account":"ACCOUNT","user":"USER","password":"PASSWORD"}

DEVICE_ID - device id

USER_ID - user id

Schedule Rules Format and Examples

rules- rules for schedules, an array type of rule

Description of the rule object:

```
public $schedule_rules = array (
    'recurrent_type' => 'no_recurrent', // no_recurrent, daily, weekly, monthly, yearly
    // flag until unappiled
    'until_unappiled' => 1,
    // corresponds to the position recurrent_type = no_recurrent.
    // If until_unappiled = 0 you need to set the interval during which
    // the policy will be active. It is set in seconds. 'until_duration' => 0,
    // !!! this field must be filled in as it acts as the start marker 'start_date' => 0, // timestamp of
    beginning date with time 00:00 'start_time' => 0, // time in seconds from the beginning of the
    start_date
    // start_date + start_time = time the schedule starts working
    // Corresponds to the position recurrent_type = daily
    // day - sets recurrence in days
    'every_days' => array (
    'day' => 1
    ),
    // corresponds to the recurrent_type = weekly
    // recur_every_week - sets recurrence in weeks from the start marker
    // days_of_week - array of included days of the week:
    array('Sun','Mon','Tue','Wed','Thu','Fri','Sat').
    // Only the ones that have been set are listed, empty stands for all days. 'every_week' => array (
    'recur_every_week' => 1,
    'days_of_week' => array()
    ),
    // Corresponds to the position recurrent_type = monthly
    // recur_flag - flag that sets recurrance type as in "every day of the week" vs "every day of the month"
    // every_day - day if recur_flag = 0
    // every_month - month if recur_flag = 0
    // recur_counter - day of the month if recur_flag = 1
    // recur_day_of_week - day of the week if recur_flag = 1
    // recur_month_counter - month if recur_flag = 1
    'every_month' => array (
        'recur_flag' => 0,
        'every_day' => 1,
        'every_month' => 1,
        'recur_counter' => 1,
        'recur_day_of_week' => 'Sun',
        'recur_month_counter' => 1
    ),
    // Relatively to position recurrent_type = yearly
    // recur_flag - flag to switch among types "every day of the year", "every day of the week of certain month"
    // every_month - month if recur_flag = 0
    // every_day - day if recur_flag = 0
    // recur_counter - date if recur_flag = 1
    // recur_day_of_week - day of the week if recur_flag = 1
    // recur_month_counter - month if recur_flag = 1 'every_yearly' =>
    array (
        'recur_flag' => 0,
        'every_day' => 1,
        'every_month' => 1,
        'recur_counter' => 1,
        'recur_day_of_week' => 'Sun',
        'recur_month_counter' => 1
    )
);
```

Example of object rules:


```
[
    {"until_duration":86400,"every_days":[{"start_time":65580,"day":1}],
"until_unapplied":"0","start_date":1331582400,"start_time":65580,"recurrent_type":"daily"}
,

{"every_week":[{"recur_every_week":"1","start_time":66000,"days_of_week":["Tue","Wed","Fri ","Sat"]}],
"until_duration":172800,"until_unapplied":"0","start_date":1331582400,"start_time":66000," recurrent_type":"weekly"},

{"until_duration":259200,"until_unapplied":"0","start_date":1331582400,"start_time":66600,
"every_month":[{"every_day":3,"every_month":"1","start_time":66600,"recur_flag":"0"}],"rec urrent_type":"monthly"},

{"until_duration":86400,"until_unapplied":"0","start_date":1331582400,"start_time":67200,"
every_yearly":[{"every_day":3,"every_month":"May","start_time":67200,"recur_flag":"0"}],"r ecurrent_type":"yearly"}
]
```


Service API Documentation

Account Service

Entity-metaname: account
Facility-metaname: fcl_account
Basic input/output object format:

```
<xml>
    <id>{id}</id>
    <name>{name}</name>
    <description>{description}</description>
    <left_key>{left_key}</left_key>
    <right_key>{right_key}</right_key>
    <level>{level}</level>
    <company_id>{company_id => [company object]}</company_id>
    <person_id>{person_id => [person object]}</person_id>
<xml>
```

Fields of the object requirements:

- name: The name may contain latin and numeric characters, spaces, and the following characters: “.”, “-”, and “_”. The name must start with an alphabetic character and then end with either a latin or numeric character. The name should have a minimum of 3 characters and a maximum of 64 characters.
- id: numeric
- description: 512 characters maximum

Table 2-1 Methods

Method	Description	Call as	Parameters
Get_childs	Gets all child objects for current account	account.get_childs	id: account ID (required)
Get_by_name	Gets account object by name	account.get_by_name	name: account name (required)

Get_top_account		account.get_top_account	none
Get_tree_by_objects	Gets account objects tree	account.get_tree_by_objects	account_id: account object ID (optional, default 0)
Create	Creates a new account object	account.create	parent_id: account parent object ID (required) object: account object (required)
Update		account.update	object: account object (required)
Delete	Deletes account object	account.delete	object: account object (required)
Registration	Creates a new account and it immediately registers the user and the device. Three input parameters of the object (account, user, and device) are the objects of the appropriate services.	account.registration	account (required) user (required) device (required)
Get_by_id	Gets account object by ID	account.get_by_id	id: account ID (required)
Get_full	Gets full information of account object including sub-objects	account.get_full	id: account ID (required)
Get_list_in	Gets account objects whose IDs are in the input array	account.get_list_in	in: account ID array (required)
Index	Gets account index	account.index	none

Address Service

Entity-metaname: address

Facility-metaname:

fcl_account Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <street_address>{street_address}</street_address>
```

```
        <city>{city}</city>
        <state_id>{state_id => [state object]}</state_id>
        <province>{province}</province>
        <zip_code>{zip_code}</zip_code>
        <country_id>{country_id => [country object]}</country_id>
    <xml>
```

Fields of the object requirements:

- street_address: 256 characters maximum
- province: 128 characters maximum
- zip_code: 16 characters maximum

Table 2-2 Methods

Method	Description	Call as	Parameters
Create	Creates a new address object	address.create	object: address object (required)
Update		address.update	object: address object (required)
Delete	Deletes address object	address.delete	object: address object (required)
Get_by_id	Gets address object by ID	address.get_by_id	id: address ID (required)
Get_full	Gets full information of address object including sub-objects	address.get_full	id: address ID (required)
Get_list_in	Gets address objects whose IDs are in the input array	address.get_list_in	in: address ID array (required)
Index	Gets address index	address.index	none

Authentication Service

System authentication service
Facility-metaname: fcl_security

Table 2-3 Methods

Method	Description	Call as	Parameters

Login		auth.login	account (required) user (required) password (required) client_name (optional, default false) client_version (optional, default false) use_experimental (optional, default false)
Is_logined		auth.is_logined	none
Logout		auth.logout	none

Company Service

Entity-metaname:

company Facility-

metaname: fcl_account

Basic input/output object

format:

```
<xml>
    <id>{id}</id>
    <name>{name}</name>
    <description>{description}</description>
    <contact_id>{contact_id => [contact object]}</contact_id>
<xml>
```

Fields of the object requirements:

Table 2-4 Methods

Method	Description	Call as	Parameters
Get_by_name	Gets company object by name	company.get_by_name	name: company name (required)
Create	Creates a new company object	company.create	object: company object (required)
Update		company.update	object: company object (required)
Delete	Deletes company object	company.delete	object: company object (required)
Get_by_id	Gets company object by ID	company.get_by_id	id: company ID (required)
Get_full	Gets full information of company object including	company.get_full	id: company ID (required)

	sub-objects		
Get_list_in	Gets company objects whose IDs are in the input array	company.get_list_in	in: company ID array (required)
Index	Gets company index	company.index	none

Contact Service

Entity-metaname: contact

Facility-metaname:

fcl_account Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <phone>{phone}</phone>
    <cell_phone>{cell_phone}</cell_phone>
    <email>{email}</email>
    <notification_url>{notification_url}</notification_url>
    <web>{web}</web>
    <skype>{skype}</skype>
    <address_id>{address_id => [address object]}</address_id>
<xml>
```

Fields of the object requirements:

- phone: 64 characters maximum
- cell_phone: 64 characters maximum
- email: 128 characters maximum
- *notification_url: 512 characters maximum*
- web: 256 characters maximum
- skype: 64 characters maximum

Table 2-5 Methods

Method	Description	Call as	Parameters
Create	Creates a new contact object	contact.create	object: contact object (required)
Update		contact.update	object: contact object (required)
Delete	Deletes contact object	contact.delete	object: contact object (required)

Get_by_id	Gets contact object by ID	contact.get_by_id	id: contact ID (required)
Get_full	Gets full information of contact object including sub-objects	contact.get_full	id: contact ID (required)
Get_list_in	Gets contact objects whose IDs are in the input array	contact.get_list_in	in: contact ID array (required)
Index	Gets contact index	contact.index	none

Country Service

Entity-metaname: country

Facility-metaname:

fcl_default Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <name>{name}</name>
<xml>
```

Table 2-6 Methods

Method	Description	Call as	Parameters
Index	Gets country index	country.index	none
Get_by_name	Gets country object by name	country.get_by_name	name: country name (required)
Get_by_id	Gets country object by ID	country.get_by_id	id: country ID (required)
Get_full	Gets full information of country object including sub-objects	country.get_full	id: country ID (required)
Get_list_in	Gets country objects whose IDs are in the input array	country.get_list_in	in: country ID array (required)

Device Service

Entity-metaname: device

Facility-metaname:

fcl_device Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <name>{name}</name>
    <mac>{mac}</mac>
    <serial_number>{serial_number}</serial_number>
    <is_disabled>{is_disabled}</is_disabled>
    <last_ip>{last_ip}</last_ip>
    <last_external_ip>{last_external_ip}</last_external_ip>
    <location>{location}</location>
    <description>{description}</description>
    <device_status>{device_status}</device_status>
    <address_id>{address_id => [address object]}</address_id>
    <account_id>{account_id => [account object]}</account_id>
    <timezone>{timezone}</timezone>
    <timezone_offset>{timezone_offset}</timezone_offset>
    <is_muted>{is_muted}</is_muted>
    <is_notlicensed>{is_notlicensed}</is_notlicensed>
    <is_updaterequired>{is_updaterequired}</is_updaterequired>
    <user_online_id>{user_online_id}</user_online_id>
<xml>
```

Fields of the object requirements:

- name: 3 characters minimum, 128 characters maximum
- description: 512 characters maximum
- mac: 12 characters maximum
- serial_number: 4 characters minimum, 32 characters maximum

Table 2-7 Methods

Method	Description	Call as	Parameters
Search		device.search	mask (required)
Mark	Marks device as online, checks device over time, and marks it as offline	device.mark	now_connected_device (required) uptime (optional, default 0) muted (optional, default false)
Get_by_serial	Gets device object by serial number	device.get_by_serial	serial: serial number (required)
Get_by_name	Gets device object by name	device.get_by_name	account_id: account object ID (required) name: device name (required)

Get_by_serial_and_nulltoken	Gets device object by serial number and null token	device.get_by_serial_and_nulltoken	serial: serial number (required)
Get_full	Gets full information of device object (including sub-objects)	device.get_full	id: device ID (required)
Index	Gets device index	device.index	account_id: account object ID (optional, default false)
Get_online		device.get_online	account_id: account object ID (optional, default false)
Get_online_count		device.get_online_count	none
Get_all_count		device.get_all_count	none
Create	Creates a new device object	device.create	object: device object (required)
Update		device.update	object: device object (required)
Delete	Deletes device object	device.delete	object: device object (required)
Effective_profile		device.effective_profile	id: device ID (required)
Effective_profile_list		device.effective_profile_list	id: device ID (required) time (optional, default false) n (optional, default 1)
Effective_policy		device.effective_policy	id: device ID (required)
Get_by_id	Gets device object by ID	device.get_by_id	id: device ID (required)
Get_list_in	Gets device objects whose IDs are in the input array	device.get_list_in	in: device ID array (required)

Device_model Service

Entity-metaname:

model Facility-

metaname: fcl_device

Basic input/output object format:

```
<xml>
    <id>{id}</id>
    <product_id>{product_id => [product object]}</product_id>
    <name>{name}</name>
    <description>{description}</description>
<xml>
```

Fields of the object requirements:

- name: 2 characters minimum, 64 characters maximum

- description: 512 characters maximum

Table 2-8 Methods

Method	Description	Call as	Parameters
Index	Gets device_model index	device_model.index	none
Create	Creates a new device_model object	device_model.create	object: device_model object (required)
Update		device_model.update	object: device_model object (required)
Delete	Deletes device_model object	device_model.delete	object: device_model object (required)
Get_by_name	Gets device_model object by name	device_model.get_by_name	product_id: product object ID (required) name: device_model name (required)
Get_by_id	Gets device_model object by ID	device_model.get_by_id	id: device_model ID (required)
Get_full	Gets full information of device_model object including sub-objects	device_model.get_full	id: device_model ID (required)
Get_list_in	Gets device_model objects whose IDs are in the input array	device_model.get_list_in	in: device_model ID array (required)

Event Service

Entity-metaname: events

Basic input/output object format:

```
<xml>
    <id>{id}</id>
    <object_type>{object_type}</object_type>
    <object_id>{object_id}</object_id>
    <event_time>{event_time}</event_time>
    <severity>{severity}</severity>
    <facility>{facility}</facility>
    <message>{message}</message>
    <data>{data}</data>
<xml>
```

Fields of the object requirements:

- object_type: 16 characters maximum
- severity: 16 characters maximum

- facility: 64 characters maximum
- message: 1024 characters maximum
- data: 4096 characters maximum

Table 2-9 Methods

Method	Description	Call as	Parameters
Register	Registers a new event object	event.register	object_type (required) object_id (required) severity (required) facility (required) message (optional, default '') data (optional, default '') time (optional, default false)
Get_facility_list		event.get_facility_list	object_type (required) object_id (required)
Index	Gets event index	event.index	object_type (optional, default 'device') object_id (optional, default 0) limit (optional, default '') severity (optional, default '') facility (optional, default '') search (optional, default '') start (optional, default 0) end (optional, default 0)
Device_perfomance_info		event.device_perfomance_info	device_id: device object ID (required) limit (optional, default '') start (optional, default 0) end (optional, default 0)
Device_perfomance_info_advanced		event.device_perfomance_info_advanced	device_id: device object ID (required) limit (optional, default '') start (optional, default 0) end (optional, default 0)
Create_notification		event.create_notification	senders (required) types (required) contains (required) receivers (required)

Get_by_id	Gets event object by ID	event.get_by_id	id: event ID (required)

Firmware Service

Entity-metaname: firmware

Facility-metaname:

fcl_device Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <version_major>{version_major}</version_major>
    <version_sys>{version_sys}</version_sys>
    <version_app>{version_app}</version_app>
    <model_id>{model_id}</model_id>
    <is_release>{is_release}</is_release>
    <is_enabled>{is_enabled}</is_enabled>
    <description>{description}</description>
    <specification_path>{specification_path}</specification_path>
    <sys_image_path>{sys_image_path}</sys_image_path>
    <app_image_path>{app_image_path}</app_image_path>
    <build>{build}</build>
    <pversion>{pversion}</pversion>
<xml>
```

Fields of the object requirements:

- name: The name may contain latin and numeric characters, spaces, and the following characters: “.”, “-”, and “_”. The name must start with an alphabetic character and then end with either a latin or numeric character.
- sys_image_path: 2048 characters maximum
- app_image_path: 2048 characters maximum
- specification_path: 2048 characters maximum
- description: 512 characters maximum

Table 2-10 Methods

Method	Description	Call as	Parameters
To_release_all		firmware.to_release_all	none
Support_index		firmware.support_index	none
Support_versions		firmware.support_versions	model_id (required)
Index	Gets firmware index	firmware.index	where (optional, default '')
Create	Creates a new firmware object	firmware.create	object: firmware object (required)

Update		firmware.update	object: firmware object (required)
Delete	Deletes firmware object	firmware.delete	object: firmware object (required)
Disable		firmware.disable	id: firmware ID (required)
Enable		firmware.enable	id: firmware ID (required)
Get_by_id	Gets firmware object by ID	firmware.get_by_id	id: firmware ID (required)
Get_full	Gets full information of firmware object including sub-objects	firmware.get_full	id: firmware ID (required)

Group Service

Account group management

service Entity-metaname:

group

Facility-metaname: fcl_account

Table 2-11 Methods

Method	Description	Call as	Parameters
Get_full	Gets full information of group object including sub-objects	group.get_full	type (required) group_id: group object ID (required)
Get_by_name	Get group object by name	group.get_by_name	account_id: account object ID (required) type (required) name: group name (required)
Get_by_id	Gets group object by ID	group.get_by_id	type (required) group_id: group object ID (required)
Get_tree_by_objects	Gets group objects tree	group.get_tree_by_objects	type (required) account_id: account object ID (required) return_as_tree (optional, default true)

Create	Creates a new group object	group.create	type (required) account_id: account object ID (required) name: group name (required) description (optional, default '') parent_id: group parent object ID (optional, default false)
Update		group.update	type (required) group_id: group object ID (required) name: group name (optional, default false) description (optional, default false)
Delete	Deletes group object	group.delete	type (required) group_id: group object ID (required)
Get_childs	Gets all child objects for current group	group.get_childs	type (required) group_id: group object ID (required)
Get_all_childs		group.get_all_childs	type (required) group_id: group object ID (required)
Add_object		group.add_object	type (required) group_id: group object ID (required) object_id (required)
Set_objects	If the is_reset parameter is set, then the group will be cleared of prior entities	group.set_objects	type (required) list_group_id (required) object_id (required) is_reset (optional, default 0)
Remove_object		group.remove_object	type (required) group_id: group object ID (required) object_id (required)
Object_not_member_of		group.object_not_member_of	type (required) group_id: group object ID (required)

Object_member_of		group.object_member_of	type (required) object_id (required) return_as_tree (optional, default true)

Image Service

Table 2-12 Methods

Method	Description	Call as	Parameters
Get_favicon_by_url		image.get_favicon_by_url	url (required)
Image_file_resize		image.image_file_resize	filedata (required) toWidth (required) toHeight (required)
Image_file		image.image_file	filedata (required)
Image_base64_resize		image.image_base64_resize	base64encodedImage (required) toWidth (required)

Notification Service

Entity-metaname:

notification Basic

input/output object format:

```
<xml>
    <id>{id}</id>
    <name>{name}</name>
    <ts>{ts}</ts>
    <account_id>{account_id => [account object]}</account_id>
    <period>{period}</period>
    <severity_facility>{severity_facility}</severity_facility>
    <message_contains>{message_contains}</message_contains>
    <rules>{rules}</rules>
    <messages>{messages}</messages>
    <logic>{logic}</logic>
<xml>
```

Fields of the object requirements:

- name: 3 characters minimum, 64 characters maximum

Table 2-13 Methods

Method	Description	Call as	Parameters
Specification		notification.specification	none

Add_user		notification.add_user	notification_id: notification object ID (required) user_id: user object ID (required)
Set_user_list		notification.set_user_list	notification_id: notification object ID (required) user_id_array (required)
Set_notification_list		notification.set_notification_list	user_id: user object ID (required) notification_id_array (required)
Remove_user		notification.remove_user	notification_id: notification object ID (required) user_id: user object ID (required)
Notification_user_list		notification.notification_user_list	notification_id: notification object ID (required)
User_notification_list		notification.user_notification_list	user_id: user object ID (required)
Create	Creates a new notification object	notification.create	account_id: account object ID (required) name: notification name (required) types (required) - see the “Arrays” section below message_contains (required) rules (required) - see the “Arrays” section below period (required) or_logic (optional, default 0)
Update	(See the notification.create method)	notification.update	id: notification ID (required) name: notification name (required) types (required) - see the “Arrays” section below message_contains (required) rules (required) - see the “Arrays” section below period (required) or_logic (optional, default 0)
Delete	Deletes notification	notification.delete	object: notification object (required)
Get_by_id	Gets notification object by ID	notification.get_by_id	id: notification ID (required)

Get_by_name	Gets notification object by name	notification.get_by_name	account_id: account object ID (required) name: notification name (required)
Index	Gets notification index	notification.index	account_id: account object ID (required)
Close_opened_notifications		notification.close_opened_notifications	none

Arrays

In the Create method (see the table above), the following parameters are arrays:

1. types - An array of settings under which Severity and Facility the notification will fire. The format is an array of objects with two parameters - severity and facility:

   [{"severity":"ERROR","facility":"*"},{"severity":"*","facility":"Co"}]

   severity: ERROR, WARNING, INFO, etc.

   facility: from output of the event::get_facility_list Both can be replaced with asterisks to include all errors:

   {"severity":"ERROR","facility":"*"}

   Or replaced with "Co" for any messages from Cobalt (the communication module):

   {"severity":"*","facility":"Co"}

2. rules - An array of thresholds. Data for this array is contained in the Specification method (notification.specification) of this service.

   [{"name":"cpu-avg","operator":">",
   "value":"60"},{"name":"cpu-memory","operator":">","value":"50"},...]

Person Service

Entity-metaname: person

Facility-metaname:

fcl_account Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <title>{title}</title>
    <first_name>{first_name}</first_name>
    <middle_name>{middle_name}</middle_name>
    <last_name>{last_name}</last_name>
    <contact_id>{contact_id => [contact object]}</contact_id>
    <company_id>{company_id => [company object]}</company_id>
```

<xml>

Fields of the object requirements:

- title: 60 characters maximum
- first_name: 50 characters maximum
- middle_name: 50 characters maximum
- last_name: 50 characters maximum

Table 2-14 Methods

Method	Description	Call as	Parameters
Create	Creates a new person object	person.create	object: person object (required)
Update		person.update	object: person object (required)
Delete	Deletes person object	person.delete	object: person object (required)
Get_by_id	Gets person object by ID	person.get_by_id	id: person ID (required)
Get_full	Gets full information of person object including sub-objects	person.get_full	id: person ID (required)
Get_list_in	Gets person objects whose IDs are in the input array	person.get_list_in	in: person ID array (required)
Index	Gets person index	person.index	none

Policy Service

Policy management service Entity-metaname: policy

Facility-metaname: fcl_profile

Table 2-15 Methods

Method	Description	Call as	Parameters
Copy		policy.copy	account_id_des (required) policy_id_list (required) new_name (optional, default false)

Duplicate		policy.duplicate	policy_id: policy object ID (required) new_name (required)
Create	Creates a new policy object	policy.create	account_id: account object ID (required) policy_name (required) property_list (required) - see "Array" section below policy_description (optional, default '') is_action (optional, default 0) is_menu_set (optional, default 0)
Update	(See policy.create method)	policy.update	policy_id: policy object ID (required) name: policy name (optional, default false) description (optional, default false) is_action (optional, default false) is_menu_set (optional, default false)
Get	Gets policy object	policy.get	policy_id: policy object ID (required)
Get_full	Gets full information of policy object including sub-objects	policy.get_full	policy_id: policy object ID (required)
Get_by_name	Gets policy object by name	policy.get_by_name	account_id: account object ID (required) name: policy name (required)
Get_tree		policy.get_tree	policy_id: policy object ID (required)
Get_list		policy.get_list	policy_id: policy object ID (required)
Get_list_in	Gets policy objects whose IDs are in the input array	policy.get_list_in	in: policy ID array (required)
Get_index		policy.get_index	account_id: account object ID (required)
Add	(See policy.create method)	policy.add	policy_id: policy object ID (required) property_list (required)
Set	(See policy.create method)	policy.set	policy_id: policy object ID (required) property_list (required)
Delete	Deletes policy object	policy.delete	policy_id: policy object ID (required)
Check_depends		policy.check_get_list	policy_id: policy object ID (required)

Device_get_list		policy.device_get_list	device_id: device object ID (required)
Device_set_list		policy.device_set_list	device_id: device object ID (required) policy_list (required) schedule_list (optional, default null)
Group_get_list		policy.group_get_list	type (required) group_id: group object ID (required)
Group_set_list		policy.group_set_list	type (required) group_id: group object ID (required) policy_list (required) schedule_list (optional, default null)
Index	Gets policy index	policy.index	none

Array

In the Create method (see the table above), the property-list parameter is an array:

- property_list - An array of objects.

  {"name of spec

  property":"value"} For

  example:

  [{"browser.url":"cnn.com"},{"display.rotation":"0"}]

Product Service

Entity-metaname: product

Facility-metaname:

fcl_device Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <name>{name}</name>
    <description>{description}</description>
<xml>
```

Fields of the object requirements:

- name: The name may contain latin and numeric characters, spaces, and the following characters: “.”, “-”, and “_”. The name must start with an alphabetic character and then end with either a latin or numeric character.
- description: 512 characters maximum

Table 2-16 Methods

Method	Description	Call as	Parameters
Index	Gets product index	product.index	none
Create	Creates a new product object	product.create	object: product object (required)
Update		product.update	object: product object (required)
Delete	Deletes product object	product.delete	object: product object (required)
Get_by_name	Gets product object by name	product.get_by_name	name: product name (required)
Get_by_id	Gets product object by ID	product.get_by_id	id: product ID (required)
Get_full	Gets full information of product object including sub-objects	product.get_full	id: product ID (required)
Get_list_in	Gets product objects whose IDs are in the input array	product.get_list_in	in: product ID array (required)

Profile Service

Profile management service Entity-metaname: profile Facility-metaname: fcl_profile

Table 2-17 Methods

Method	Description	Call as	Parameters
Index	Gets profile index	profile.index	data (required)
Default_copy		profile.default_copy	account_id_des (required) profile_id_list (required)
Default_create		profile.default_create	account_id: account object ID (required) profile_name (required) property_list (required) profile_description (optional, default

			'')
Default_update		profile.default_update	profile_id: profile object ID (required) name: profile name (optional, default false) description (optional, default false)
Default_get		profile.default_get	profile_id: profile object ID (required)
Default_get_full		profile.default_get_full	profile_id: profile object ID (required)
Default_get_by_name		profile.default_get_by_name	account_id: account object ID (required) profile_name (required)
Default_tree		profile.default_tree	profile_id: profile object ID (required)
Default_list		profile.default_list	profile_id: profile object ID (required)

Default_index		profile.default_index	account_id: account object ID (required)
Default_get_list_in		profile.default_get_list_in	in: profile ID array (required)
Default_add		profile.default_add	pprofile_id: profile object ID (required) pproperty_list (required)
Default_set		profile.default_set	pprofile_id: profile object ID (required) pproperty_list (required)
Default_delete		profile.default_delete	profile_id: profile object ID (required)
User_tree		profile.user_tree	user_id: user object ID (required)
User_list		profile.user_list	user_id: user object ID (required) use_attaching (optional, default false)

User_create		profile.user_create	user_id: user object ID (required) default_id (required)
User_add		profile.user_add	user_id: user object ID (required) property_list (required)
User_set		profile.user_set	user_id: user object ID (required) property_list (required)
Device_tree		profile.device_tree	device_id: device object ID (required)
Device_list		profile.device_list	device_id: device object ID (required) use_attaching (optional, default false)
Device_create		profile.device_create	device_id: device object ID (required) default_id (required)
Device_add		profile.device_add	device_id: device object ID (required) property_list (required)

Device_set		profile.device_set	device_id: device object ID (required) property_list (required)
Get_by_id	Gets profile object by ID	profile.get_by_id	id: profile ID (required)
Get_full	Gets full information of profile object including sub-objects	profile.get_full	id: profile ID (required)
Get_list_in	Gets profile objects whose IDs are in the input array	profile.get_list_in	in: profile ID array (required)

Role Service

Entity-metaname: role

Facility-metaname:

fcl_security Basic

input/output object format:

```
<xml>
    <id>{id}</id>
    <title>{title}</title>
    <name>{name}</name>
    <description>{description}</description>
<xml>
```

Fields of the object requirements:

- name: The name may contain latin and numeric characters, spaces, and the following characters: “.”, “-”, and “_”. The name must start with an alphabetic character and then end with either a latin or numeric character. The name should have a maximum of 64 characters.
- title: 128 characters maximum
- description: 512 characters maximum

Table 2-18 Methods

Method	Description	Call as	Parameters
Get_by_name	Gets role object by name	role.get_by_name	name: role name (required)
Create	Creates a new role object	role.create	object: role object (required)
Update		role.update	object: role object (required)
Delete	Deletes role object	role.delete	object: role object (required)
Get_by_id	Gets role object by ID	role.get_by_id	id: role ID (required)
Get_full	Gets full information of role object including sub-objects	role.get_full	id: role ID (required)
Get_list_in	Gets role objects whose IDs are in the input array	role.get_list_in	in: role ID array (required)
Index	Gets role index	role.index	none

Schedule Service

Entity-metaname:

schedule Basic

input/output object

format:

```
<xml>
    <id>{id}</id>
    <account_id>{account_id => [account object]}</account_id>
    <name>{name}</name>
    <rules>{rules}</rules>
    <description>{description}</description>
<xml>
```

See the “Schedule Rules Format and Examples” section of this chapter. Fields of the object requirements:

1. name: The name may contain latin and numeric characters, spaces, and the following characters: “.”, “-”, and “_”. The name must start with an alphabetic character and then end with either a latin or numeric character. The name should have a minimum of 3 characters and a maximum of 64 characters.
2. description: 512 characters maximum.

Table 2-19 Methods

Method	Description	Call as	Parameters

Get_simple		schedule.get_simple	id: schedule ID (required)
Get_by_name	Gets schedule object by name	schedule.get_by_name	account_id: account object ID (required) name: schedule name (required)
Index	Gets schedule index	schedule.index	account_id: account object ID (optional, default false)
Get_list		schedule.get_list	none
Create	Creates a new schedule object See the “Schedule Rules Format and Examples” section.	schedule.create	object: schedule object (required)
Update	See the “Schedule Rules Format and Examples” section.	schedule.update	object: schedule object (required) See the “Schedule Rules Format and Examples” section.
Delete	Deletes schedule object	schedule.delete	object: schedule object (required)
Unset_device_policy		schedule.unset_device_policy	device_id: device object ID (required) policy_id: policy object ID (required)
Set_device_policy		schedule.set_device_policy	device_id: device object ID (required) policy_id: policy object ID (required) schedule_id: schedule object ID (required)
Unset_group_policy		schedule.unset_group_policy	group_id: group object ID (required) policy_id: policy object ID (required)
Set_group_policy		schedule.set_group_policy	group_id: group object ID (required) policy_id: policy object ID (required) schedule_id: schedule object ID (required)
Assign_to		schedule.assign_to	schedule_id: schedule object ID (required)

Processing_device_list		schedule.processing_device_list	device_id: device object ID (required) time (optional, default 0) count (optional, default 1
Processing_device		schedule.processing_device	device_id: device object ID (required) time (optional, default 0) return_policy_id_only (optional, default false)
Get_by_id	Gets schedule object by ID	schedule.get_by_id	id: schedule ID (required)

Specification Service

Specification control service
Entity-metaname: specification
Facility-metaname: fcl_profile

Table 2-20 Methods

Method	Description	Call as	Parameters
Get	Gets specification object	specification.get	type (optional, default null) use_experimental (optional, default false)
Get_list		specification.get_list	type (optional, default null) use_experimental (optional, default false)
Get_actions		specification.get_actions	account_id: account object ID (required)
Get_menu_actions		specification.get_menu_actions	account_id: account object ID (required)
Set_action_policy		specification.set_action_policy	device_id: device object ID (required) policy_id: policy object ID (required)
Set_action_property		specification.set_action_property	device_id: device object ID (required) action_name (required) action_value (required)

Set_action_property_list	action_list is always an array of objects {"name of the action property":"value"} For example: [{"power.reboot":"1"},{"display.rotation":"0"}]	specification.set_action_property_list	device_id: device object ID (required) action_list (required)
Set_action_group_policy		specification.set_action_group_policy	account_id: account object ID (required) group_id: group object ID (required) policy_id: policy object ID (required)
Set_action_group_property		specification.set_action_group_property	account_id: account object ID (required) group_id: group object ID (required) action_name (required) action_value (required)
Set_action_group_property_list	action_list is always an array of objects {"name of the action property":"value"} For example: [{"power.reboot":"1"},{"display.rotation":"0"}]	specification.set_action_group_property_list	account_id: account object ID (required) group_id: group object ID (required) action_list (required)

System Service

Table 2-21 Methods

Method	Description	Call as	Parameters
Import_device_csv		system.import_device_csv	account_id: account object ID (required) csv_file (required)
Import		system.import	account_id: account object ID (required) xml (required) is_validation (optional, default false)
All_export		system.all_export	none

All_import		system.all_import	filename (required)
Bulk_export	Input is an array of objects {entity:id}. entity : 'user', 'device', 'profile', 'policy', 'notification', 'schedule' id - identifiers For example: [{"device":"32"},{" device":"39"},{"use r":"89"}] Returns a link to the file that can be downloaded.	system.bulk_export	objects (required)
Account_export		system.account_export	account_id: account object ID (required)
Export		system.export	entity (required) id: system ID (required)
Change_log		system.change_log	data (required)
Change_log_types		system.change_log_types	none
Change_log_operati on_types		system.change_log_operatio n_types	none
Changes		system.changes	none
Maintainance_code		system.maintainance_code	serial (required)
Preferences_load		system.preferences_load	none
Preferences_save	Saves preferences. Input is an array of objects {property_name:val ue}	system.preferences_save	preferences (required)
Changes_log		system.changes_log	for_last_time (optional, default 0)
Changes_log_full		system.changes_log_full	for_last_time (optional, default 0)
Check_license		system.check_license	license_string (required)
Get_licenses		system.get_licenses	none
Add_license		system.add_license	license_string (required)
Remove_license		system.remove_license	license_string (required)
Mac_address_list		system.mac_address_list	none
Admin_warning		system.admin_warning	none
Testing		system.testing	none
Search		system.search	mask (required)

User Service

Entity-metaname:

user Facility-

metaname: fcl_user

Basic input/output object format:

```
<xml>
    <id>{id}</id>
    <account_id>{account_id => [account object]}</account_id>
    <title>{title}</title>
    <login>{login}</login>
    <passwd>{passwd}</passwd>
    <description>{description}</description>
    <person_id>{person_id => [person object]}</person_id>
    <role_id>{role_id => [role object]}</role_id>
    <temp_code>{temp_code}</temp_code>
    <lock_time>{lock_time}</lock_time>
    <failed_attempts>{failed_attempts}</failed_attempts><xml>
<xml>
```

Fields of the object requirements:

- login: 3 characters minimum, 64 characters maximum
- description: 512 characters maximum

Table 2-22 Methods

Method	Description	Call as	Parameters
Index	Gets user index	user.index	account_id: account object ID (optional, default false)
Full_index		user.full_index	account_id: account object ID (optional, default false)
Get_by_login	Gets user object by login	user.get_by_login	account_id: account object ID (required) login (required)
Get_full	Gets full information of user object including sub-objects	user.get_full	id: user ID (required)
Create	Creates a new user object	user.create	object: user object (required)
Change_password		user.change_password	id: user ID (required) oldpassword (required) newpassword (required)
Update		user.update	object: user object (required)
Delete	Deletes user object	user.delete	object: user object (required)
Get_by_id	Gets user object by ID	user.get_by_id	id: user ID (required)
Get_list_in	Gets user objects whose IDs are in the input array	user.get_list_in	in: user ID array (required)

Widgets

Chapter Overview

In addition to regular plugins support (like Adobe Flash Player), Cobra provides several proprietary widgets to simplify developer's life. Those widgets can be configured and controlled using JavaScript.

The topics in this chapter include:

- Plugins API
- Camera Widget
- DrmViewer Widget
- Jabber Client Framework (JCF) Widget
- RemoteDesktop Widget
- SipPhone Widget
- Ticker Widget
- VideoPlayer Widget
- VncViewer Widget
- WebBrowser Widget
- WebClip Widget
- WebRTC Widget

Plugins API

Every plugin inherits the following interface:

```
interface Plugin
        {
        readonly attribute int status ; signals

        :

        void statusChanged(in int status) ; void
        ready();
        void error(in string message);
        };
```

- status: Contains current plugin status.
  - 0: Ready to be used
  - 1: In progress, which usually means that the initialization process in not finished yet
  - 2: Error

- statusChanged(): Fires when plugin's status is changed.
- ready(): Fires when plugin becomes ready (i.e. plugin's status changed to 0).
- error(): Fires when an error occurs (i.e. plugin's status changed to 2).

Tip It is a good practice to always check a plugin's status before you use it. It is especially important when you use network-related plugins, which load their content from network, because those plugins perform asynchronously. See the VideoPlayer plugin usage example for an example of such a good practice.

Camera Widget

The Camera plugin allows you to access an USB camera connected to the IEC4600 Series. This plugin allows you to make shots and get those shots as JPEG images. It is highly recommended to use an USB video class (UVC) camera.

Note Camera hot plugging is supported by the application, but hot unplugging is not.

The Camera interface declaration is the following:

```
interface Camera
{
        readonly attribute string errorString; readonly
        attrbute string toJpeg; readonly attrbute bool
        ready; attribute bool controlsVisible; attribute
        int delay;
        attribute string cameraDevice; readonly
        attribute int camerasFound;

parameters:
        string cameraDevice;
        int delay;
        bool constrolsVisible ;

slots:

        void discover();
        void shoot();
        void setEnabled(in bool enabled);

signals:
        void captured();
        void error();
        void discovered(in int numberOfCameras);
        void discovered();

};
```

Note Starting with 2.2.1, the following variables should not be used: cameraDevice, camerasFound, discover(). Instead use global.system.webCameras to get the map with the found cameras and their associated devices.

Table 3-1 Variables

Variable	Description
cameraDevice	Specifies the system device associated with the selected camera. **Note** Use global.system.webCameras to get the map with the found cameras and their associated devices.
camerasFound	The number of found cameras. **Note** This variable is deprecated. Use global.system.webCameras.
cameraDevice	This parameter is used to specify the low level device to be used for the camera object.
delay	This parameter is used to specify the delay in seconds before shooting.
constrolsVisible	This parameter is used to show controls.
discover()	Use this to discover cameras. **Note** This variable is deprecated. Use global.system.webCameras.

The following HTML document contains an example of camera usage (file.../.../test/plugins/camera.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
    <title>..:: camera plugin test ::.. </title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<s
tyl
e>
bo
dy
{
    margin: 0;
    background-color: #000000;
    font-weight: bold;
    font-family: Arial;
    font-size: 20px;
    color :
    #eeeeee;
}
    </style>
    <script type="text/javascript"> var
    camera;
    var image;
    var
    camerasFound;
    function init ()
{
```

```
        camera = document.getElementById("camera"); image
        = document.getElementById("image");
        camerasFound = document.getElementById("camerasFound");

        // image has been captured
        camera.captured.connect(captured);

        // error occured
        camera.error.connect(error) ;

        // discovering has finished , use camera.camerasFound to read
        // the number of cameras found
        camera.discovered.connect(discovered);
}
function captured()
{
        // camera.toJpeg doesn't cache images, so 3 calls to 'camera.toJpeg'
        // will read image data 3 times
        image.src = "" + camera.toJpeg;
}
function error ()
{
        // display error string alert
        (camera.errorString);
}
function discovered()
{
        // camera.camerasFound == -1 if not initialized camerasFound.innerHTML = "Cameras
        found: " + camera.camerasFound;
}
function isReady()
{
        alert (camera.ready);
}
function setEnabled(enab)
{
        camera.setEnabled(enab);
}

function showAvailable()
{
        var map = global.system.webCameras; var
        result = "";
        // key = camera device (for example "/dev/video0")
        // value = camera's
        human
        •                                                                                •
        readably name (for example "HD-1000 USB video camera") for (var i in
        map)
        {
                result += i + ": " + map[i];
        }
        alert ( result ) ;
}
        function setDevVideo(index)
        {
                // you can also assign keys from the map retrieved from global.system.webCameras,
                // see showAvailable()
                camera.cameraDevice = "/dev/video" + index;
        }

</script>
```

```
</head>
    <body onload="init()">
        <center>
        <a href='#' onclick="showAvailable()">Show available</a>
        <a href='#' onclick="setDevVideo(0)">/dev/video0</a>
        <a href='#' onclick="setDevVideo(1)">/dev/video1</a>
        <a href='#' onclick="alert(camera.cameraDevice)">Current camera</a>

        <!-- deprecated, use global.system.webCameras instead (bear in mind backward compability) -->
        <a href='#' onclick="camera.discover()">Discover</a>
        <a href='#' onclick="camera.shoot()">Shoot</a>
        <a href='#' onclick="isReady()">Ready?</a>
        <a href='#' onclick="setEnabled(true)">Enable</a>
        <a href='#' onclick="setEnabled(false)">Disable</a>

        <div id="camerasFound" style="text-align: center;">
        </div>

        <br/>

        <div>
        <object id="camera" type="application/x-qt-plugin" classid="camera" width="50%" height="40%">
        <param name="delay" value="4" /> <!-- delay before shooting -->
        <param name="constrolsVisible" value="true" /> <!-- show controls -->
        </object>
        </div>

        <br/>

        <div id="result">
        <img id="image" src="" width="640" height="480" />
        </div>
        </center>
    </body>
</html>
```

DrmViewer Widget

The DrmViewer plug-in plays Digital Rights Management (DRM) content from a Swank media player. The widget includes APIs for Shaka-player related trick mode operations and has full screen toggle on/off without viewing interruptions.

The interface declaration is the following:

```
interface DrmViewer
{
    attribute string url ;
methods:
    void runJavaScript(in string scriptSource) ;
    variant evaluateJavaScript(in string scriptSource) ; parameters:
    string url
; signals :
        void loadStarted();
        void loadFinished(in bool ok); void
        loadProgress(in int percent); void
        urlChanged(in url url) ;
        void titleChanged(in string title ) ;
};
```

Table 3-2 Variables

Variable	Description
runJavaScript	Executes the specified JavaScript code *scriptSource* in drmviewer. The JS code is executed asynchronously.
evaluateJavaScript	Executes the specified JavaScript code *scriptSource* in drmviewer. The JS code is executed synchronously and its result is returned.
url	URL pointing to an HTML document.
useragent	HTTP User-Agent.

The following HTML document contains an example of drmviewer usage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
                <title>..:: drmviewer plugin test ::.. </title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <s
    tyl
    e>
    bo
    dy
    {
        margin: 0;
        background-color: #000000;
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
        color : #eeeeee;
    }
    </style>
    <script type="text/javascript"> var
    addressBar;
    var
    webBrowser;
    function init ()
    {
        addressBar = document.getElementById("addressBar");
        webBrowser = document.getElementById("webBrowser");
        addressBar.value = webBrowser.url;
    }
    function deinit ()
    {
        webBrowser.evaluateJavaScript("terminate();");
    }
    function go()
    {
    webBrowser.url = addressBar.value;
    }
    function msg()
    {
        webBrowser.runJavaScript("alert('Hello from inside drmviewer');");
    }
    function changeSize()
    {
        webBrowser.style. left = "20px";
        webBrowser.style.top = "120px";
```

```
            webBrowser.style.width = "800px";
            webBrowser.style.height = "480px";
        }
        function runJSSync()
        {
            alert (webBrowser.evaluateJavaScript("returnInteger();"));
        }
        </script>
        </head>
        <body onload="init()" onbeforeunload="deinit()">
        <input id="addressBar" type="text" />
        <button onclick="go()">Go</button>
        <button onclick="msg()">Message</button>
        <button onclick="changeSize()">Change size</button>
        <button onclick="runJSSync()">Get a JS result from drmviewer</button>
        <br />
        <br />
        <font
        color="#FFFFFF">WebEngine
        •                                                                    •
        based browser</font>
        <div style="text-align: center ;">
        <object id="webBrowser" type="application/x-qt-plugin" classid="drmviewer" style="position:absolute; left: 0px;
            top: 100px; width: 640px; height: 400px; border: 3px solid #00FF00">
            <param name="url" value="http://127.0.0.1:8000/test/plugins/drmcontent.html"/>
        </object>
        </div>
        </body>
</html>
```

Jabber Client Framework (JCF) Widget

JCF is the native implementation of Cisco Jabber Client Framework. This widget offers functionality similar to the SipPhone widget, except that JCF is Cisco proprietary implementation. This plugin enables the application to perform like a feature-rich SIP endpoint and supports a wide range of audio and video codecs such as g711, g722, g722.1, g729 for audio and H.264, H.263, H.261 for video codecs respectively. The Cisco H.264 implementation is open sourced and is compatible with most H.264 decoders. If your deployment uses many Cisco endpoints, it is recommended that you to use this widget instead of the SipPhone widget.

```
interface Jcf
{
        attribute string status; attribute
        string idleImage;

parameters:
        string

camera; slots:

        int start (in string domain,
                        in string username, in
                        string password, in
                        string tftpAddr);
        void call(in string sipUri,
                        in bool withVideo);
        void hangup();
        int callHold(); int
        callResume();
        int setVolume(in int vol); int
        mute();
        int unmute();
        bool isRegistered();
```

```
        void sendDtmf(in string dtmfkey); string
        cameraDevice();
        int setCameraDevice(in string cameraName); int
        answer(in bool videoFlag);
        int reject();
        bool setidleImage(in string imgurl, in bool stretchFlag); bool changeidleImage(in
        string imgurl, in bool stretchFlag); bool addVideoToCall(in bool videoFlag);

signals:

        void idle();
        void incomingCall(in string peer); void
        placingCall();
        void ring();
        void trying();
        void established();
        void disconnected();
        void registered();
        void telemetry(in string callStats); void
        hold();
        void resume();
        void
        novideo();
        void video();
        void error(in int code, in string message);
};
```

Table 3-2 JCF Variables includes the description of the methods used for various features. Additionally, it lists the signals available in the system.

Table 3-3 JCF Variables

Variable	Description
camera	Use this method to specify the name of the camera to be used for the call.
start(in string domain, in string username, in string password)	Use this method to set the JCF credentials that are needed to get registered with the CUCM specified by the variable domain, either as an IP address or as a FQDN. You can use this credential to register as a Cisco endpoint with the UCM.
call(in string sipUri, in bool withVideo)	This method should be used only after the start(...) method is called. This method initiates the call to SIP URI (Called party). Note The second parameter, in bool withVideo, is optional and is set to True, by default. So, if you do not specify the parameter, it will initiate a video call. You can set the second argument as False to make only audio calls.
hangup()	This method disconnects the ongoing call.
callHold()	This method puts the remote caller on hold. The

	remote caller will stay on hold unless the callResume() method is called.
callResume()	This method will resume the call on hold.
setVolume(in int vol)	This method enables you to set/control the volume of the application. The advantage of JCF over SipPhone is that it provides application level volume control.
mute()	This method mutes the microphone. It prevents transmission of any audio output.
unmute()	This method is used to unmute the microphone and start transmitting the audio output.
isRegistered()	Based on whether the endpoint is registered or not, this method returns Boolean True or False, to make and receive calls.
sendDtmf(in string dtmfkey)	This method sends Dual Tone Multiple Frequencies (DTMF) as RFC2833 to the other end. Valid DTMF keys are 0-9, *, and #.
cameraDevice()	This method returns the configured webcam that is currently being used by the JCF widget.
setCameraDevice(in string cameraName)	Use this method to let the JCF widget know which webcam to use while placing the call.
answer(in bool withVideo)	Use this method to accept an incoming call. The 'withVideo' parameter is optional. By default, it is set to True. If you would like to have only audio call, set "Audio only' parameter as 'False'.
reject()	This method is used to reject the incoming call.
setidleImage(in string imgUrl, in bool stretchFlag)	This method can be used to display an image such as a logo or some graphic when the SIP widget is registered and not on a call. The parameters are imgUrl, the URL for the image to be displayed, and stretchFlag, which indicates whether to auto-resize the image to the given frame.
changeidleImage(in string imgUrl, in bool stretchFlag)	This method has similar functionalities as setidleImage. This method can be used to change the appearance of the widget, like coding it in the javascript to change the idleimage to create a screen saver for the widget.
addVideoToCall(in bool videoFlag)	This method enables you to add or remove the video stream from an existing call. In order to use this method, the call should be negotiated as a video call. Set the function as: • True - To enable the video stream in the call Or • False - To disable the video stream. By default, the video stream is enabled.
Signals	Description
ready()	It indicates that the values given for initializing the

	JCF is accepted.
registered()	It means that the JCF is now registered with the SIP Registrar (or Call Manager) and you can make/receive calls from the widget.
placingCall()	The widget is trying to make a call to the desired party.
incomingCall()	The widget is receiving an incoming call from another SIP peer.
established()	It indicates that the call is established.
ring()	This signal means that the called party has been notified about the incoming call.
disconnected()	The call has been terminated.
telemetry(in string callStats)	This signal contains the complete details of the previous disconnected call. Use this to display as regular Telemetry data.
video()	This signal means that a video call is initiated and the remote site/caller video is available to display.
novideo()	This signal means that the video is not sent by the remote end for the call. The application can use video/novideo signal to enhance the user experience. For example, displaying a message like Please wait....
hold()	The remote caller has put the call on hold.
resume()	The remote caller has resumed the call.
error(in int code, in string explanation)	This signal is indicative of any errors while using the widget. The signal has an error code along with the explanation/details of the error.

The Following HTML document contains an example showing how to use jcf.

```
<!DOCTYPE html>

 <html>
 <head>
 <link rel="stylesheet" type="text/css" href="css/time.css">
 <link rel="stylesheet" type="text/css" href="css/button.css">
 <link rel="stylesheet" type="text/css" href="css/modal.css">
 <link rel="stylesheet" type="text/css" href="css/keypad.css">
 <script type="text/javascript" src="js/alert.js"></script>

 <s
 tyl
 e>
 bo
 dy
 {
       background-image: url('img/CiscoBlack.png');
       background-repeat: no-repeat;
       background-attachment: fixed;
       background-position: center;
 }

 .frame {
```

```
        float: left ; width:49.5%;
        height:100%;
        overflow:auto;
    }
    .frame .frame {
        background:cyan
    }
    .frame.border {
        border-right:3px ridge buttonface }
    p.normal {
        font-style: normal;

    }

    p.  italic {
        font-style: italic ;
    }

    p.oblique {
        font-style: oblique;
        color: white;
        text-align: center;
    }

    div {
        font-style: oblique;
        color: white;
        text-align: center;
        vertical-align: center;
    }

    statusLabel {
        font-style: oblique;
        font-size: larger;
        color: white;
        text-align: left ;
        top:30px;
        vertical-align: center;
    }

    h2 {
        color: #1d1d1d;
    }

    a:visited {
        color:
        #fff;
    }

    a:link {
        color: #fff;
    }

    a:active {
        color: #fff;
}

    #m1 {
        position:absolute;
        width:20;
        height:20;
        bottom:80px;
```

```
        right:25%;
        left:50%;
        margin-left:-330px;
}

#m2 {
        position:absolute;
        width:20; height:20;
        bottom:80px;
        right:25%;
        left:50%;
        margin-left:-230px;
}
#m3 {
        position:absolute;
        width:20; height:20;
        bottom:80px;
        right:25%;
        left:50%;
        margin-left:-130px;
}
#m4 {
        position:absolute;
        width:20;
        height:20;
        bottom:80px;
        right:25%;
        left:50%;
        margin-left:-30px;
}
#m5 {
        position:absolute;
        width:20;
        height:20;
        bottom:80px;
        right:25%;
        left:50%;
        margin-left:+70px;
}
#m6 {
        position:absolute;
        width:20;
        height:20;
        bottom:80px;
        right:2
        5%;
        left:50
        %;
        margin-left:+170px;
}
#m7 {
        position:absolute;
        width:20;
        height:20;
        bottom:80px;
        right:25%;
        left:50%; margin-left:+270px;
}

videoView {
        position: fixed;
        top: 0;
```

```
        left : 0;
        right: 0;
        bottom:
        130px;
        width:
        100%;
        height:
        720px;
        overflow:
        hidden;
        background: #fff;
}

label {
        color: #B4886B;
        font-weight: bold;
        display: block;
        width: 150px;
        float: left;
}

textarea {
        border: 1px solid #888;
}

#dtmfModal {
        position: fixed;
        bottom: 30px;
        overflow:
        hidden; width:
        100%;
}
#volumeModal {
        position: absolute;
        bottom: 40px;
        overflow: hidden;
        width: 100%;
        margin-left: +110px;
}
@font-face {
        font-family: CiscoSansTTLightOblique;
        src: url(font/CiscoSansTTLightOblique.ttf); font-weight:
        bold;
}
</style>
</head>

<!-- CODE & LOGIC HERE -->
<script>
var jcfPhone = null; var
dtmfPadShow = true; var
volumeShow = true; var
micMuted = false;

var callHeld = false; var
inCall = false; var
prevSignal = "";

function init()
{
        GetClock();
        setInterval(GetClock,1000);
        //console.log("** JCF Script (mod. April-27-2016) **"); connectSlots();
```

```
        getStatus();
    }


    </script>

    <!-- MAIN CODE ENDS -->

    <body onLoad="init()">

    <!--
    <script>
    function displayDate() { var
    d = new Date();
    if (d){
        document.getElementById("demo").innerHTML = d.toString();
     }
    }
    </script>
    -->
    </p>

    <script type="text/javascript">


    tday=new Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday");
    tmonth=newArray"January","February","March","April","May","June","July","August",
    "September","October","November","December");

    function GetClock(){
    var d=new Date();
    var nday=d.getDay(),nmonth=d.getMonth(),ndate=d.getDate(),nyear=d.getYear(); if(nyear<1000) nyear+=1900;
    var nhour=d.getHours(),nmin=d.getMinutes(),nsec=d.getSeconds(),ap;

    if(nhour==0){ap=" AM";nhour=12;} else
    if(nhour<12){ap="AM";}
    else if(nhour==12){ap="PM";}
    else if(nhour>12){ap="PM";nhour-=12;}

    if(nmin<=9) nmin="0"+nmin;
    if(nsec<=9) nsec="0"+nsec;

    document.getElementById('clockbox').innerHTML=""+tday[nday]+", "+tmonth[nmonth]+" "+ndate+", "+nyear+" "+nhour+
  ":"+nmin+":"+nsec+ap+"";
    }

    </script>
    <div id="clockbox"></div>
    <div id="statusLabel"></div>

    <!--
    <div id="Avatar" align="center" class="round-button">
      <div class="round-button-circle">
        <a href="javascript:getStatus();" class="round-button">
        </a>
  </div>
</div>
    -->

    <table align=center>
      <tr>
        <td>
          <div id="m1"
            align="center" class="round-button">
```

```
        <div class="round-button-circle">
          <a href="#dialModal" class="round-button" id="phoneAction">
            <img src="img/phone.svg" id="phoneImage" width="48" height="48"></img>
          </a>
        </div>
      </div>
    </td>
    <td>
      <div id="m2" align="center" class="round-button">
        <div class="round-button-circle">
          <a href="javascript:onHoldPress();" class="round-button">
            <img src="img/pause.svg" id="pause" width="48" height="48"></img>
          </a>
        </div>
      </div>
    </td>
    <td>
      <div id="m3" align="center" class="round-button">
        <div class="round-button-circle">
          <a href="javascript:onDtmfPad();" class="round-button">
            <img src="img/dial.svg"></img>
          </a>
        </div>
      </div>
    </td>
    <td>
      <div id="m4" align="center" class="round-button">
        <div class="round-button-circle">
          <a href="#statsModal" class="round-button">
            <img src="img/stats.png" width="48" height="48"></img>
          </a>
        </div>
      </div>
    </td>
    <td>
      <div id="m5" align="center" class="round-button">
        <div class="round-button-circle">
          <a href="javascript:onVolumePress();" class="round-button">
            <img src="img/speaker.png" id="speaker" width="48" height="48"></img>
          </a>
        </div>
      </div>
    </td>
    <td>
      <div id="m6" align="center" class="round-button">
        <div class="round-button-circle">
          <a href="javascript:onMicPress();" class="round-button">
            <img src="img/microphone.png" id="mic" width="48" height="48"></img>
          </a>
        </div>
      </div>
    </td>
    <td>
      <div id="m7" align="center" class="round-button">
        <div class="round-button-circle">
          <a href="#configModal" class="round-button">
            <img src="img/config.png"></img>
          </a>
        </div>
      </div>
    </td>
  </tr>
```

```
</table>
  <!-- Configuration Dialog -->
  <div id="configModal" class="modalDialog">
    <div>
      <a href="#close" title="Close" class="close">X</a>
      <h2>Configure JCF</h2>
      <table>
        <tr>
        <td>
          <h3>CUCM IP / Name</h3>
        </td>
        <td>
          <input type="text" name="CucmIP" id="CucmIP">
        </td>
      </tr>
      <tr>
        <td>
          <h3>CUCM UserName</h3>
        </td>
        <td>
          <input type="text" name="CucmUser" id="CucmUser">
        </td>
      </tr>
      <tr>
        <td>
          <h3>CUCM Password</h3>
        </td>
        <td>
          <input type="password" name="CucmPass" id="CucmPass">
        </td>
      </tr>
    </table>
      <input type="image" src="img/save.png" onclick="saveAndApply()" width="60%" >
    </div>
  </div>

  <!-- Status Window Starts --> <div id="statsModal" class="statsDialog">
    <div>
      <a href="#close" title="Close" class="close">X</a>
        <h2>Call Telemetry</h2>
          <textarea id="callStats" rows="20" cols="80"></textarea>
      </div>
    </div>

  <!-- Status Window Ends -->
    <script>
    function onDtmfPad() {
      // Dont do anything
      if we are not In a call if (inCall != true) return;

      if (dtmfPadShow == true)
    {
              dtmfPadShow = false; document.getElementById("dtmfModal").style.display = 'block';
    }
    else
    {
              dtmfPadShow = true; document.getElementById("dtmfModal").style.display =
              'none';
    }
  }

  function onVolumePress() { if
```

```
    (volumeShow == true) {
   volumeShow = false; document.getElementById("volumeModal").style.display = 'block';


   }
   else
   {


   volumeShow = true; document.getElementById("volumeModal").style.display = 'none';
   }
   }
 function onMicPress() { if
 (micMuted == false)
    {
micMuted = true;
document.getElementById("mic").src = "img/mutedmic.png"; var jcfPhone =
document.getElementById("jcf");
//console.log("Calling MUTE : "); jcfPhone.mute();


}
else
{


micMuted = false;
document.getElementById("mic").src = "img/microphone.png"; var jcfPhone =
document.getElementById("jcf");
//console.log("Calling UNMUTE : ");
jcfPhone.unmute();

}
}
 function onHoldPress() {
    // Dont do anything if we are not in a Call if (inCall !=
    true) return;

    var jcfPhone = document.getElementById("jcf"); if (callHeld
    == false)
    {
callHeld = true;
document.getElementById("pause").src = "img/accept.png";
//console.log("Calling CALL HOLD");
jcfPhone.callHold();

}
}



callHeld = false;
document.getElementById("pause").src = "img/pause.svg";
//console.log("Calling CALL RESUME");
jcfPhone.callResume();

function appendDial(num) {
   document.getElementById("DIAL").value += num;
   }

   function trimDial() {
         var cur = document.getElementById("DIAL").value; cur =
         cur.substring(0, cur.length - 1);
         document.getElementById("DIAL").value = cur;
```

```
}

function onCallButton() {
        var c = document.getElementById("DIAL").value; var jcf =
        document.getElementById("jcf");
        //console.log("NUMBER IS " + c);
        jcf.call(c, true);


        document.getElementById("phoneImage").src = "img/conversation_end_button.png";
        document.getElementById("phoneAction").href = "javascript:endCall();";
        inCall = true;
        location.href="#close";
}

function saveAndApply() {
      var  cucmIP  =  document.getElementById("CucmIP").value; var cucmUser
      = document.getElementById("CucmUser").value; var cucmPass =
      document.getElementById("CucmPass").value;
      //console.log("CUCM IP " + cucmIP);
      //console.log("CUCM USR " + cucmUser);
      //console.log("CUCM PASS " + cucmPass);
      saveToApplicationData();
      var jcfPhone = document.getElementById("jcf");
      jcfPhone.start(cucmIP, cucmUser, cucmPass);
      location.href="#close";
}

function endCall() {
      var jcfPhone = document.getElementById("jcf"); jcfPhone.hangup(); inCall = false;


      document.getElementById("phoneImage").src = "img/phone.svg"; document.getElementById("phoneAction").href =
      "#dialModal";
}

function acceptCall() {
      location.href="#close";
      var jcfPhone = document.getElementById("jcf");
      jcfPhone.answer();
      document.getElementById("phoneImage").src = "img/conversation_end_button.png";
      document.getElementById("phoneAction").href = "javascript:endCall();";
      inCall = true;
}

function rejectCall() {
     location.href="#close";
     var jcfPhone = document.getElementById("jcf");
     jcfPhone.reject(); document.getElementById("phoneImage").src = "img/phone.svg";
     document.getElementById("phoneAction").href = "#dialModal";
     inCall = false;
}

function holdCall(toggle) {
      var jcfPhone = document.getElementById("jcf"); if (toggle) { jcfPhone.hold();
      showHideInCallButton(false);
   } else { jcfPhone.resume();
      showHideInCallButton(false);
   }
}
function showHideInCallButton(flag) {
     /*
      if (flag)
```

```
        {
            // SHOW
            //console.log("SHOWING IN CALL BUTTON"); document.getElementById("m1").style.display = 'none';
            document.getElementById("m3").style.display = 'none'; document.getElementById("m5").style.display = 'block';
            document.getElementById("m6").style.display = 'none';
        }
        else
        {
            // HIDE
            //console.log("HIDING IN CALL BUTTON"); document.getElementById("m1").style.display = 'block';
            document.getElementById("m3").style.display = 'block'; document.getElementById("m5").style.display = 'none';
            document.getElementById("m6").style.display = 'none';
        }
       */
}

function getStatus() {
      var jcfPhone = document.getElementById("jcf"); var status =
      jcfPhone.status;
      if (!status)
      {
                  status = " initializing ... ";
      }
        document.getElementById("statusLabel").innerHTML = status;
        //console.log("STATUS is " + status);
}

function connectSlots() {
        var jcf = document.getElementById("jcf");
        jcf.ready.connect(onReady);
        jcf.incomingCall.connect(onIncomingCall);
        jcf.placingCall.connect(onPlacingCall);
        jcf.ring.connect(onRing);
        jcf.trying.connect(onTrying);
        jcf.established.connect(onEstablished);
        jcf.disconnected.connect(onDisconnected);
        jcf.registered.connect(onRegistered);
        jcf.hold.connect(onHold);
        jcf.resume.connect(onResume);
        jcf.novideo.connect(onNoVideo);
        jcf.video.connect(onVideo);
        jcf.error.connect(onError);
        jcf.telemetry.connect(onTelemetry);
}

function onReady() {
        //console.log("OnReady Received"); document.getElementById("statusLabel").innerHTML = "Ready";
}

function onIncomingCall(peer) {
        // Dont react to multiple Firing of same signal if (prevSignal
        == 'incoming')
        return;

        prevSignal = 'incoming';

        //console.log("Incoming Call From Peer : " + peer); document.getElementById("statusLabel").innerHTML = "Incoming
        Call From " + peer; location.href ="#incomingModal";
}

function onPlacingCall() {
        //console.log("Calling ..."); document.getElementById("statusLabel").innerHTML = "Calling...";
```

```
}

function onRing() {
	//console.log("Ringing..."); document.getElementById("statusLabel").innerHTML = "Ringing";
}

function onTrying() {
	//console.log("Trying..."); document.getElementById("statusLabel").innerHTML = "Trying";
}

function onEstablished() {
	//console.log("Call Established ..."); document.getElementById("statusLabel").innerHTML = "Established";
	setTimeout(function(){
			document.getElementById("statusLabel").innerHTML = "In Call";}, 5000);
}
function onDisconnected() {
	//console.log("Call Disconnected...");
	location.href ="#close";
	document.getElementById("statusLabel").innerHTML = "Disconnected"; prevSignal = "";
	setTimeout(function(){
			document.getElementById("statusLabel").innerHTML = "Ready";}, 5000);
}

function onRegistered() {
	//console.log("Registered Successfully ... "); document.getElementById("statusLabel").innerHTML = "Registered
	Successfully"; setTimeout(function(){
			document.getElementById("statusLabel").innerHTML = "Ready";}, 5000);
}

function onHold() {
	 //console.log("Call Held ...."); document.getElementById("statusLabel").innerHTML = "Call Held";

}

function onResume() {
	 //console.log("Call Resumed...."); document.getElementById("statusLabel").innerHTML = "Call Resumed";
	 setTimeout(function(){
			document.getElementById("statusLabel").innerHTML = "Ready";}, 5000);
}

function onVideo() {
	//console.log("On Video Signal...");
	var l = document.getElementById("statusLabel").innerHTML;
	document.getElementById("statusLabel").innerHTML = "Video"; setTimeout(function(){
			document.getElementById("statusLabel").innerHTML = l;}, 5000);
}


function onNoVideo() {
	//console.log("On No Video Signal...");
	var l = document.getElementById("statusLabel").innerHTML;
	document.getElementById("statusLabel").innerHTML = "Video"; setTimeout(function(){
			document.getElementById("statusLabel").innerHTML = l;}, 5000);
}

function onError(code, msg) {
	//console.log("ERROR MSG : " + msg); document.getElementById("statusLabel").innerHTML = "ERROR:" + code
	+ msg;
}

function onSendDTMF(key) {
   var jcfPhone = document.getElementById("jcf");
   //console.log ("SENDING DTMF With Key " + key);
   jcfPhone.sendDtmf(key);
```

```
}

function onTelemetry(msg) {
        document.getElementById("callStats").innerHTML = msg; location.href
        ="#statsModal";
        //console.log ("CALL TELEMETRY : " + msg);
}

</script>
<!-- Incoming Call Dialog -->
<div id="incomingModal" class="modalDialogSmall">
  <div>
  <a href="#close" title="Close" class="close">X</a>
  <h2>Incoming Call</h2>
  <br>
  <div id="peerCalling"></div>
  <br>
  <br>
  <script>
        document.getElementById('peerCalling').innerHTML =
        document.getElementById('statusLabel').innerHTML;
  </script>
  <input type="image" src="img/accept.png" onclick="acceptCall()" width="64" height="64">
  <input type="image" src="img/exit.svg" onclick="rejectCall()" width="64" height="64">
  </div>
</div>

<!-- Dial Pad -->
<div id="dialModal" class="modalDialogSmall">
<div>
  <a href="#close" title="Close" class="close">X</a>
  <h2>Dial Pad</h2>
  <label>
  <table align="center">
    <tr>
      <td><input type="text" name="DIAL" id="DIAL"> </td>
  <td>
        <input type="image" src="img/delete.svg" width="24" height="24" onclick="trimDial()">
  </td>
    </tr>
  </table>
  </label>

  <table align="center">
    <container>
      <tr>
      <td onclick="appendDial('1')">
            <a href="#dialModal"><block>1</block></a>
      </td>
      <td onclick="appendDial('2')">
            <a href="#dialModal"><block>2</block></a>
      </td>
      <td onclick="appendDial('3')">
            <a href="#dialModal"><block>3</block></a>
      </td>
      </tr>
      <tr>
      <td onclick="appendDial('4')">
            <a href="#dialModal"><block>4</block></a>
      </td>
      <td onclick="appendDial('5')">
            <a href="#dialModal"><block>5</block></a>
      </td>
```

```
        <td onclick="appendDial('6')">
            <a href="#dialModal"><block>6</block></a>
        </td>
        </tr>
        <tr>
        <td onclick="appendDial('7')">
            <a href="#dialModal"><block>7</block></a>
        </td>
        <td onclick="appendDial('8')">
            <a href="#dialModal"><block>8</block></a>
        </td>
        <td onclick="appendDial('9')">
            <a href="#dialModal"><block>9</block></a>
        </td>
        </tr>
        <tr>
        <td onclick="appendDial('*')">
            <a href="#dialModal"><block>•</block></a>
        </td>
        <td onclick="appendDial('0')">
            <a href="#dialModal"><block>0</block></a>
        </td>
        <td onclick="appendDial('#')">
            <a href="#dialModal"><block>#</block></a>
        </td>
        </tr>
      </container>
</table>
    <input type="image" src="img/accept.png" onclick="onCallButton()" width="64"height="64">
  </div>
 </div>

 <div id="frame">
  <div id="frame border">
    <object id="jcf" type="application/x-qt-plugin" classid="jcf" camera="TANDBERG" width="1280px" height="720px">
    </object>
  </div>
 </div>

 <!-- DTMF Pad -->
 <div id="dtmfModal" align="center" style="display:none;">
  <div>
    <table bgcolor="#cdcdcd" align="center">
    <tr>
    <td onclick="onSendDTMF('1')">
        <a href="#dtmfModal"><block1>1</block1></a>
    </td>
    <td onclick="onSendDTMF('2')">
        <a href="#dtmfModal"><block1>2</block1></a>
    </td>
    <td onclick="onSendDTMF('3')">
        <a href="#dtmfModal"><block1>3</block1></a>
    </td>
    <td onclick="onSendDTMF('4')">
        <a href="#dtmfModal"><block1>4</block1></a>
    </td>
    <td onclick="onSendDTMF('5')">
        <a href="#dtmfModal"><block1>5</block1></a>
    </td>
    <td onclick="onSendDTMF('6')">
        <a href="#dtmfModal"><block1>6</block1></a>
    </td>
```

```
        <td onclick="onSendDTMF('7')">
            <a href="#dtmfModal"><block1>7</block1></a>
        </td>
        <td onclick="onSendDTMF('8')">
            <a href="#dtmfModal"><block1>8</block1></a>
        </td>
        <td onclick="onSendDTMF('9')">
            <a href="#dtmfModal"><block1>9</block1></a>
        </td>
        <td onclick="onSendDTMF('0')">
            <a href="#dtmfModal"><block1>0</block1></a>
        </td>
        <td onclick="onSendDTMF('*')">
            <a href="#dtmfModal"><block1>•</block1></a>
        </td>
        <td onclick="onSendDTMF('#')">
            <a href="#dtmfModal"><block1>#</block1></a>
        </td>
          </tr>
        </table>
      </div>
    </div>

    <!-- Speaker Volume START -->
    <div id="volumeModal" align="center" style="display:none;">
<script>
        function updateSlider(slideAmount) {
            var sliderDiv = document.getElementById("sliderAmount"); var jcfPhone =
            document.getElementById("jcf"); sliderDiv.innerHTML = slideAmount;
            setTimeout(function(){
                                volumeShow =
                                false;
                                onVolumePress();
                              }, 4000);
            if (slideAmount == 0)
            {
                document.getElementById("speaker").src = "img/mutespeaker.png";
            }
            else
            {
                document.getElementById("speaker").src = "img/speaker.png";
            }
                //console.log("CALLING setVolume with : " + slideAmount);
                jcfPhone.setVolume(slideAmount);

        }
    </script>
    <input id="slide" type="range" min="0" max="100" step="1" value="75" onchange="updateSlider(this.value)">
    <div id="sliderAmount"> </div>
    </div>

    <!-- Speaker Volume END -->

    <!-- DEBUG PURPOSE ONLY START -->
    <script>
    function saveToApplicationData() { var
        cucmIP;
        var
        cucmUser;
        var
        cucmPass;
        cucmIP = document.getElementById("CucmIP").value; cucmUser =
        document.getElementById("CucmUser").value; cucmPass =
```

```
        document.getElementById("CucmPass").value;

        // Check? global.applicationData.insert("sip.domain", cucmIP);
        global.applicationData.insert("sip.username", cucmUser); global.applicationData.insert("sip.password", cucmPass);

        // Camera?
}

setTimeout(function(){
                        var temp = "default";
                        var cucmIP = global.applicationData.value("sip.domain", temp);
                        var cucmUser = global.applicationData.value("sip.username", temp); var cucmPass =
                        global.applicationData.value("sip.password", temp);

                        if ((cucmIP == temp) || (cucmUser == temp) || (cucmPass == temp))
                        {
                                alert("Please Configure Application Data via IEM"
                                        + " or via the Application's Configure button ");
                        }
                        else
                        {
                                document.getElementById("CucmIP").value = cucmIP;
                                document.getElementById("CucmUser").value = cucmUser;
                                document.getElementById("CucmPass").value = cucmPass;
                                saveAndApply();
                           }
                        }, 5000);
</script>
<!-- DEBUG PURPOSE ONLY END -->
</div>
</body> </html>
```

PdfViewer Widget

The PdfViewer plugin provides the ability to display Adobe PDF documents on the browser.

```
interface PdfViewer
 {

        attribute string url;
        attribute int dpi;

        attribute bool controlsVisible; attribute
        bool pageNumberVisible; attribute int
        pageNumber;
        readonly attribute int numberOfPages;

        int next();
        int previous();

      signals:

            void loadStarted();
            void loadFinished(in bool ok);

};
        attribute bool controlsVisible; attribute
        bool pageNumberVisible; attribute int
        pageNumber;
        readonly attribute int numberOfPages;

        int next();
```

```
    int previous();

signals:

        void loadStarted();
        void loadFinished(in bool ok);

};
```

The following HTML document contains an example of Pdf Viewer usage (file ../../test/plugins/pdfviewer.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>

        <title>..:: pdfviewer plugin test ::..</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<s

tyl

e>

bo

dy
{
        margin: 0;
        background-color: #000000;
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
        color: #eeeeee;
}

</style>

<script type="text/javascript"> var

pdfViewer;
var textDpi;
var checkBoxControlsVisible; var
checkBoxPageNumberVisible;

function main()
{
        textDpi.value = pdfViewer.dpi; checkBoxControlsVisible.checked =
        pdfViewer.controlsVisible;
        checkBoxPageNumberVisible.checked = pdfViewer.pageNumberVisible;
}
function error(message)
{
        alert ("PDF viewer error: " + message);
}

function init()
{
        pdfViewer = document.getElementById("pdfViewer");
        textDpi = document.getElementById("textDpi");
        checkBoxControlsVisible = document.getElementById("checkBoxControlsVisible"); checkBoxPageNumberVisible =
```

```
        document.getElementById("checkBoxPageNumberVisible");

        pdfViewer.loadFinished.connect(main);
        pdfViewer.error.connect(error);
    }

    function setControlsVisible()
    {
        pdfViewer.controlsVisible = checkBoxControlsVisible.checked;
    }

    function setPageNumberVisible()
    {
        pdfViewer.pageNumberVisible = checkBoxPageNumberVisible.checked;
    }

    function setDpi()
    {
        pdfViewer.dpi = textDpi.value;
    }

    </script>

    </head>

    <body onload="init()">

        <div align="center">
        DPI <input id="textDpi" type="text" />
        <input type="button" value="Set DPI" onclick="setDpi();" />   
        <input type="button" value="Previous page" onclick="pdfViewer.previous();" />
        <input type="button" value="Next page" onclick="pdfViewer.next();" />
        <br />
        <object id="pdfViewer" type="application/x-qt-plugin" classid="pdfviewer" width="480" height="640">
        <param name="url" value="http://embedded-computing.com/pdfs/QNX.Jan05.pdf"/>
        <param name="dpi" value="150"/>
        <param name="controlsVisible" value="true"/>
        <param name="pageNumberVisible" value="true"/>
        <param name="pageNumber" value="3"/>
        </object>

        <br />
        <input id="checkBoxControlsVisible" type="checkbox" onchange="setControlsVisible()" />Controls visible
        <input id="checkBoxPageNumberVisible" type="checkbox" onchange="setPageNumberVisible()" />Page number
        visible
        </div>

    </body>

     </html>
```

RemoteDesktop Widget

```
interface RemoteDesktop
{
    readonly attribute bool isConnected;
    attribute bool interactive ;

methods:
    void connectToHost(in string host, in int port, in string password, in bool interactive = true,
        in string serverPublicKey = ""); void
```

```
        disconnect() ;
        string errorString (in int code);

signals :
        void error(in int code); void
        connected();
        void disconnected();
};
```

Table 3-4 Variables

Variable	Description
interactive	Whether the user can control the remote desktop with mouse and keyboard. Even if interactive is true, the server can decline the user activity events.
connectToHost	Connect to the given host and port with the given password. If *serverPublicKey* is not empty, it will be compared to the public key of the server. If it does not match, the connection will not be established. *serverPublicKey* must be generated using the RSA algorithm and encoded in the PEM format.
disconnect	Disconnect from the host.
errorString	Returns the error description.

Note This plugin uses a proprietary protocol and is only compatible with the Remote Desktop Server.

The following HTML contains an example of remotedesktop usage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
        <head>
                <title>..:: remotedesktop plugin test ::.. </title>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <s
        tyl
        e>
        bo
        dy
        {
                margin: 0;
                background-color: #000000;
                font-weight: bold;
                font-family: Arial;
                font-size: 20px;
                color : #eeeeee;
        }
        </style>
        <script type="text/javascript"> var
        host;
        var   port;
        var
        password;
        var interactive ;
```

```
var rd;
var status
; var key;
function connected()
{
	status .innerHTML = "<font color='#00FF00'>CONNECTED</font>";
}
function disconnected()
{
	status .innerHTML = "<font color='#FF0000'>DISCONNECTED</font>";
}
function error(code)
{
	alert ("Remote desktop error: " + rd.errorString(code));
}
function init ()
{
	host = document.getElementById("host"); port =
	document.getElementById("port");
	password = document.getElementById("password"); interactive =
	document.getElementById("interactive"); key =
	document.getElementById("key");
	rd = document.getElementById("rd");
	status = document.getElementById("status");
	rd.connected.connect(connected);
	rd.disconnected.connect(disconnected);
	rd. error .connect(error) ;
	disconnected();
}
function connect()
{
	// host, port, password, interactive , public key to check (can be empty) rd.connectToHost(host.value,
	parseInt(port.value) , password.value, interactive
	.checked, key.value) ;
}
function disconnect()
{
	rd.disconnect() ;
}
function isConnected()
{
	alert (rd.isConnected ? "Yes" : "No");
}
function interactiveChanged(obj)
{
	rd. interactive = obj.checked;
}
</script>

</head>

<body onload="init()">

<div align="center">
	Host: <input id="host" type="text" />
	: <input id="port" type="text" value="6050" /> Password:
	<input id="password" type="text" />
	<label><input id="interactive" type="checkbox"onclick='interactiveChanged(this);'
	/>Interactive</label>
	<br />
	Public key to check:
	<br />
```

```
            <textarea id="key" rows="4" cols="64" style="font-family:Bitstream Vera Sans Mono,Courier New,monospace;">
            </textarea>
            <br />
            <button onclick="connect()">Connect</button>
            <button onclick="disconnect()">Disconnect</button>
            <button onclick="isConnected()">Connected?</button>
            <br />
            <object id="rd" type="application/x-qt-plugin" classid="remotedesktop"width="640" height="480">
            </object>
            <br />
            <span id="status"></span>
        </div>
        </body>
</html>
```

SipPhone Widget

The SipPhone widget allows you to make SIP video calls to another SIP endpoint.

This plugin acts like a True SIP endpoint and supports both audio and video calls. Both SD (g711) and HD (g7221) audio codecs are supported. For video, it supports H.263 and H.264 codecs.

```
interface SipPhone
{
        attribute int height;
        attribute int width;
        attribute string backgroundColor;
        attribute string idleImage;
        attribute bool videoEnabled; // Is true by default. attribute string
        status;

parameters:
        int
        width;
        int
        height;
        int backgroundColor;
        string camera; string
        resolution ; string
        bitrate ; string fps ;
        slots:

        int start (in string username, in
                string password,
                in string domain,
                in string transport); void
        call(in string sipUri); void
        hangup();
        void sendDtmf(in string dtmfkey);
        bool setidleImage(in string imgurl, in bool stretchFlag); bool changeidleImage(in
        string imgurl, in bool stretchFlag);

        bool addVideoToCall(in bool videoFlag); bool
        enableAudibleTones(in bool toneFlag); string
        cameraDevice() const;
        int setCameraDevice(in string deviceId); int capture()
        const;
        string getImage() const; // Returns the Jpeg imgage if captured
```

```
        void
        answer();
        void reject();
        void setAutoAnswer(in bool autoAnswerFlag); signals:

        void ready();
        void registered();
        void placingCall();
        void incomingCall();
        void established();
        void ring();
        void disconnected();
        void video();
        void novideo();
        void hold();
        void resume();
        void
        captured();
        void error(in int code, in string explanation);
}
```

Table 3-5 SipPhone Parameters

Parameters	Description
width	Use this parameter to define the width for the Sip widget.
height	This parameter is used to define the height for the Sip widget.
backgroundColor	Use this parameter to define the background color of the Sip widget.
camera	This parameter is used to specify the USB webcam to be used. This should be in the Unix Device format; like, /dev/video0.
resolution	This parameter could be used when you want Sip widget to send the video at a specified resolution. **Note** If you choose higher resolution like 1280x720 or 1920x1080, the CPU utilization will increase. Additionally, the resolution should be in the format of WxH. If WxH is not in the Aspect Binding Ratio, SipPhone will try to match the video encode to the closest possible resolution.
bitrate	Use this parameter to specify the minimum bitrate that the Sip Widget should use when sending the encoded frame.
fps	Usr this parameter to specify the minimum fps (Frames Per Second) that should be captured for the video encoding.

Table 3-6 SipPhone Variables

Variable	Description
start(in string username, in string password, in string domain, in string transport)	This method call is to be used to set the SIP credentials that are needed to get registered with the SIP Registrar (or Call Manager). The needed credentials are Username, Password, Domain (IP Address or Domain Name of the SIP Registrar) and the transport to be used (UDP or TCP).
call(in string sipUri)	This method should be used only after the start(...) method is called. This method initiates the call to the sipUri (called party).
hangup()	This method, when called, disconnects the existing call.
sendDtmf(in string dtmfkey)	This method sends DTMF tones to the SIP proxy. Valid DTMF keys are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, and #.
setidleImage(in string imgUrl, in bool stretchFlag)	This method can be used to display an image, like logo or some graphic when the SIP widget is registered and not in a call. This method provides a mechanism for the widget to display an image when it is not in a call. The parameters are imgUrl, the URL for the image to be displayed, stretchFlag, which indicates whether to auto resize or not the image to the given frame.
changeidleImage(in string imgUrl, in string sipUri)	This method is similar in functionality to setidleImage. You could use this method to change the appearance of the widget like coding it in javascript to change the idleimage to create the sense of screen saver for the widget.
addVideoToCall(in bool videoFlag)	This method enables to either add or remove the video stream from an existing video call. The prerequisite is that the call should be negotiated as a video call. • Set this function as True to enable the video stream in the call. Or • Set this function with videoFlag as False to disable the video stream from the video call. By default, video stream is enabled.
enableAudibleTones(in bool toneFlag)	If you would like to enable the default audible alerts for all the call events, set the parameter of this API as True. By default, audible alerts are disabled.
cameraDevice()	This method returns the currently configured webcam that is being used by the SipPhone widget. The value returned would be in the UNIX format similar to “/dev/video0”.

setCameraDevice()	Use this method to let the SipPhone widget know which webcam to use to place the call. You need to call this API with UNIX format identifier for camera, such as "/dev/video0" or "/dev/video1".  Note Call this API before the start() method in the Javascript.
capture()	Use this method to initiate taking a still image when the video call is in progress. This is useful if you would like to take a snapshot of the participant and save it for future reference.  Caution Call this routine only when there is an active video call.
getImage()	Call this method after you have received the captured() signal. When this method is called, the routine returns base64 content of the captured JPEG image.
answer()	Accepts incoming call.
reject()	Rejects incoming call.
setAutoAnswer(in bool autoAnswerFlag)	Enables auto answer mode if the autoAnswerFlag is "true".
ready()	This signal is indicative that values given for initializing the sipphone are accepted.
registered()	This signal means that the sipphone is now registered with the SIP Registrar (or Call Manager) and you can make and receive calls from the widget.
placingCall()	This signal means the widget is trying to place the call to the called party of interest.
incomingCall()	This signal means the widget is receiving an incoming call request from another SIP peer.
established()	This signal is indicative that the call is in progress.
ring()	This signal means that the called party has been notified about the incoming call.
disconnected()	This signal means that the call has been terminated.
video()	This signal means that the call was negotiated as a video call and the remote site video is available to display.
novideo()	This signal means that the call that was negotiated does not have video being sent by the

	remote end. The application can use the novideo signal to improve the user experience such as displaying a “Please wait” message.
hold()	This signal means that the remote party has put the call on hold. An image can be displayed on the screen when a SIP call is placed on hold. The image that is included in this signal will be shown on the screen.
resume()	This signal means that the remote party has resumed the call. Upon receiving this signal, the application will revert to the original screen and the on-hold image will be hidden.
captured()	This signal is fired when an user action of taking a still snapshot of an video call is successfully finished. Once this signal is fired, the application can call getImage() to get the captured image (as a JPEG image).
error(in int code, in string explanation)	This signal is indicative of any errors whilst the widget operation. The signal has a code and an explanation about the error that was encountered. Error Codes: 404: No answer 401: Registration failed 485: User busy 494: User not found 486: Call manager not able to route call 503: Service unavailable

The following HTML document contains an example of SipPhone usage.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>

      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>..:: sipphone test ::..</title>

 <style type="text/css">

 html,
 body
 {
        padding: 0;
        margin: 0;
        width: 100%;
        height: 100%
 }
```

```
body
{
	background:
	#1e2024; color:
	#ffffff;
	font: normal 12px Arial, Helvetica, sans-serif;
}

.cameraDisabled,
.cameraReady,
.cameraOnline,
.cameraError
{
	width:
	44px;
	height:
	24px;
	background: transparent url('images/webcam-DISABLED.png') center center no-repeat;
}
.cameraReady
{
	background-image: url('images/webcam-READY.png');
}

.cameraOnline
{
	background-image: url('images/webcam-ONLINE.png');
}

.cameraError
{
	background-image: url('images/webcam-OFFLINE.png');
}

.topPanel
{
	background: #0e1014 url('images/bg-top-panel.png') top left repeat-x; color: #4c5058;
	text-shadow: 0 -1px 1px #000;
	font: normal 14px Arial, Helvetica, sans-serif; height: 41px;
}

.status
{
	color: #848d9d;
	text-shadow: 0 -1px 1px #000;
	font: normal 14px Arial, Helvetica, sans-serif; padding: 0
	20px;
}

.bottomPanel
{
	background: #0e1014 url('images/bg-bottom-panel.png') top left repeat-x; color: #4c5058;
	text-shadow: 0 -1px 1px #000;
	font: normal 14px Arial, Helvetica, sans-serif; height: 80px;
}

.calltime
{
	color: #b1b6c3;
	font: normal 14px Arial, Helvetica, sans-serif; height: 41px;
```

```
        text-shadow: 0 -1px 1px #000;
}

.buttons
{
        text-align: center; display:
        inline-block; padding: 0
        30px 0 30px;
}

.callButton,
.endCallButton,
.acceptButton,
.rejectButton
{
        width:
        170px;
        height:
        56px;
        background: transparent url('images/call-IDLE.png') center center no-repeat; border: none;
}

.callButton:hover
{
        background-image: url('images/call-HOVER.png');
}

.callButton:active
{
        background-image: url('images/call-PRESSED.png');
}

.callButton:disabled
{
        background-image: url('images/call-DISABLED.png');
}

.endCallButton
{
        background-image: url('images/call-end-IDLE.png');
}

.endCallButton:hover
{
        background-image: url('images/call-end-HOVER.png');
}

.endCallButton:active
{
        background-image: url('images/call-end-PRESSED.png');
}

.endCallButton:disabled
{
        background-image: url('images/call-end-DISABLED.png');
}

.acceptButton
```

```
{
	background-image: url('images/accept-IDLE.png');
}

.acceptButton:hover
{
	background-image: url('images/accept-HOVER.png');
}

.acceptButton:active
{
	background-image: url('images/accept-PRESSED.png');
}
.acceptButton:disabled
{
	background-image: url('images/accept-DISABLED.png');
}

.rejectButton
{
	background-image: url('images/reject-IDLE.png');
}

.rejectButton:hover
{
	background-image: url('images/reject-HOVER.png');
}

.rejectButton:active
{
	background-image: url('images/reject-PRESSED.png');
}

.rejectButton:disabled
{
	background-image: url('images/reject-DISABLED.png');
}

.view
{
	background: #000;
	border: solid 1px #3b3d40;
	text-align: center;
	width: 100%;
}

.timerOn,
.timerOff
{
	color: #b1b6c3;
	font: normal 36px Arial, Helvetica, sans-serif; height: 41px;
	text-shadow: 0 -1px 1px #000;
}

.timerOff
{
	color: #292e33;
}
```

```
.dislpad_disabled
{
        width:
        43px;
        height:
        43px;
        background: transparent url('images/dialpad-DISABLED.png') center center no-repeat; border: none;
}

</style>

<script type="text/javascript"> var

sipphone;

// These are the Credentials for the SIP endpoint
// It is recommended that you use the
// Application Data at the IEM profile to set these
// Values and get them via the global.applicationData.value() API.

var username = global.applicationData.value("sip.username", "default"); var password =
global.applicationData.value("sip.password", "default"); var domain =
global.applicationData.value("sip.domain","default");
var transport = global.applicationData.value("sip.transport", "default");


function init()
{
      sipphone = document.getElementById("sipphone");

        // Now Call Start Routine with the SIP Credentials
        // that we got from the applicationData sipphone.start(username,
        password, domain, transport);

        sipphone.placingCall.connect(onPlacingCall);
        sipphone.incomingCall.connect(onIncomingCall);
        sipphone.ready.connect(onReady); sipphone.registered
        .connect(onRegistered); sipphone.established
        .connect(onEstablished); sipphone.ring.connect(onRing);
        sipphone.disconnected.connect(onDisconnected);
        sipphone.novideo.connect(onNoVideo);
        sipphone.video.connect(onVideo);
        sipphone.hold.connect(onHold);
        sipphone.resume.connect(onResume); sipphone.error
        .connect(onError); }

function onNoVideo()
{
        document.getElementById("Status").innerHTML = "No Video...";
}
function onVideo()
{
        document.getElementById("Status").innerHTML = "Video Started (In Call)";
}


function onHold()
{
        document.getElementById("Status").innerHTML = "Call Held by Remote End";
}
function onResume()
{
        document.getElementById("Status").innerHTML = "Call Resumed (In Call)"; function
```

```
onPlacingCall()
{
        document.getElementById("Status").innerHTML = "Calling"; document.getElementById("CallButton").className =
        "callButton"; document.getElementById("cameraStatusLed").className = "cameraReady";
        document.getElementById("Status").innerHTML = "Calling..."; document.getElementById('theTime').className =
        "timerOn";
}

function onIncomingCall()
{
        document.getElementById("Status").innerHTML = "Incoming call"; document.getElementById("CallButton").className
        = "acceptButton"; document.getElementById("cameraStatusLed").className = "cameraReady";
        document.getElementById('theTime').className = "timerOn";
}

function onReady()
{
        document.getElementById("Status").innerHTML = "Ready"; document.getElementById("CallButton").className =
        "callButton"; document.getElementById("cameraStatusLed").className = "cameraDisabled";
        document.getElementById('theTime').className = "timerOff";
}

function onRegistered()
{
document.getElementById("Status").innerHTML = "Registered"; document.getElementById("CallButton").className =
"callButton";

        document.getElementById("cameraStatusLed").className = "cameraDisabled";
        document.getElementById('theTime').className = "timerOff";
}

function onEstablished()
{
        document.getElementById("Status").innerHTML = "In Call";
        document.getElementById("CallButton").className = "endCallButton";
        document.getElementById("cameraStatusLed").className ="cameraOnline";
        document.getElementById('theTime').className = "timerOn"; countDown(1);
}

function onRing()
{
        document.getElementById("Status").innerHTML = "Calling"; document.getElementById("CallButton").className =
        "acceptButton"; document.getElementById("cameraStatusLed").className = "cameraReady";
        document.getElementById('theTime').className = "timerOn";
}

function onDisconnected()
{
        document.getElementById("Status").innerHTML = "Ready"; document.getElementById("CallButton").className =
        "callButton"; document.getElementById("cameraStatusLed").className = "cameraDisabled";
        document.getElementById('theTime').className = "timerOff";
        countDown(0);
}

function onError(code, explanation)
{
        document.getElementById("Status").innerHTML = "<span style='color:#ff6920;'>Error :
      </span>" +
      explanation + " (SIP code = " + )"; document.getElementById("CallButton").className =
        "callButton"; document.getElementById("cameraStatusLed").className = "cameraError";
```

```
}

var callInProgress = false;

function call(uri)
{
    if (callInProgress)
      {
      callInProgress = false;
      sipphone.hangup();
      }
      else
      {
      callInProgress = true;
      sipphone.call(uri);
      }
}

// Conversation time.
var sec = 0;
var min =
0; var hrs
= 0;
var conversationTimer;

function stopConversationTimer()
{
      window.clearTimeout(conversationTimer); sec = 0;
      min = 0;
}

function tickConversationTimer()
{
      ++nse;
      if (sec == 59)
      {
      sec = 00;
      ++min;
      }
      if (min == 59)
      {
      min = 00;
      ++hrs;
      }
      if (sec <= 9) sec
      = "0" + sec;

      time = (hrs <= 9 ? "0" + hrs : hrs) + ":" + (min <= 9 ? "0" + min : min) + ":" + sec;
      document.getElementById('theTime').innerHTML = time;
      conversationTimer = window.setTimeout("tickConversationTimer();", 1000);
}

</script>

</head>

<body onload="init()">
      <table border="0" cellpadding="0" cellspacing="0" width="100%" height="100%">
      <tr>
      <td align="center">
      <table border="0" cellpadding="0" cellspacing="1" align="center">
```

```
  <tr>
  <td class="topPanel">
  <table border="0" cellpadding="0" cellspacing="0" width="100%" height="100%" style="width: 100%; max-width:
640px">
  <tr>
<td width="49%" style="min-width: 160px" align="left"><span class=" status"
id="Status"> </span></td>
  <td align="center">
  <div style="min-width: 162px">
  <table border="0" cellpadding="0" cellspacing="0" height="100%">
  <tr>
  <td width="44"><div style="width: 44px"> </div></ td>
  <td width="74" align="center" style="width: 74px"><img id="camera"
src="images/eye.png" width="30"
height="30" /></td>
  <td width="44"><img id="cameraStatusLed" class="
cameraDisabled" src="images/spacer.png" width="44 "
height="24" /></td>
  </tr>
  </table>
  </div>
  </td>
  <td width="49%" style="min-width: 160px" align="right"> </td>
  </tr>
  </table>
  </td>
  </tr>
  <tr>
  <td class="view">
  <object id="sipphone" type="application/x-qt-plugin" classid="sipphone" width="640" height="480">
  <param name="url" value="http://"/>
  <param name="width" value="640"/>
  <param name="height" value="480"/>
  </object>
  </td>
  </tr>
  <tr>
  <td class="bottomPanel">
  <table border="0" cellpadding="0" cellspacing="0" width="100%" height=" 100%" style="width:
100%; max-width: 640px">
  <tr>
  <td align="right" width="49%" style="min-width: 160px"><span class= "timerOff"
id="theTime">0:00</span></td>
  <td align="center">
  <div class="buttons">
  <input type="button" id="CallButton" class="callButton" value=""
onclick="call('sip:130');" />
  </div>
  </td>
  <td width="49%" style="min-width: 160px; text-align: right">
  <div style="display: inline-block; padding: 0 20px; width: 80%; text
-align: left;">
  <input type="button" value="" id="dialpad" class=" dialpad_disabled" title="Show
dialpad" />
  </div>
  </td>
  </tr>
  </table>
  </td>
  </tr>
  </table>
  </td>
```

```
        </tr>
        </table>
        <div style="display:none; visibility:hidden">
        <!-- Changing images. -->
        <img src="images/webcam-DISABLED.png" width="0" height="0" />
        <img src="images/webcam-READY.png" width="0" height="0" />
        <img src="images/webcam-ONLINE.png" width="0" height="0" />
        <img src="images/webcam-OFFLINE.png" width="0" height="0" />

        <img src="images/call-IDLE.png" width="0" height="0" />
        <img src="images/call-HOVER.png" width="0" height="0" />
        <img src="images/call-PRESSED.png" width="0" height="0" />
        <img src="images/call-DISABLED.png" width="0" height="0" />
      <img src="images/call-end-IDLE.png" width="0" height="0" />
        <img src="images/call-end-HOVER.png" width="0" height="0" />
        <img src="images/call-end-PRESSED.png" width="0" height="0" />
        <img src="images/call-end-DISABLED.png" width="0" height="0" />

        <img src="images/accept-IDLE.png" width="0" height="0" />
        <img src="images/accept-HOVER.png" width="0" height="0" />
        <img src="images/accept-PRESSED.png" width="0" height="0" />
        <img src="images/accept-DISABLED.png" width="0" height="0" />

        <img src="images/reject-IDLE.png" width="0" height="0" />
        <img src="images/reject-HOVER.png" width="0" height="0" />
        <img src="images/reject-PRESSED.png" width="0" height="0" />
        <img src="images/reject-DISABLED.png" width="0" height="0" />
      </div>
</body>
</html>
```

You can enter the call manager information either in this widget or in an IEM policy. If you want to hard code the call manager information in this widget, enter the information in the sipphone.start(username, password, domain, transport) line. You will need the call manager ID, which is a number, for the username, the call manager password, and the IP address for the domain. Enter “udp” for the transport.

See Appendix F of the *Moderro Interactive Experience Client 4600 Series User Guide* for detailed instructions on how to configure a SIP client.

Ticker Widget

The Ticker plugin allows you to create a ticker displaying a scrolling text. The native ticker widget's text size is limited to 1024 characters.

```
interface Ticker
{
      attribute string text; attribute int
      fontSize; attribute string
      fontFamily; attribute bool
      fontBold; attribute bool
      fontItalic;
      attribute string backgroundImage;
      attribute string backgroundColor;
      attribute string textColor; attribute
      string direction; attribute int speed;

parameters:
      string text ;
      int fontSize ;
```

```
        string fontFamily;
        bool fontBold;
        bool fontItalic ;
        string backgroundImage;
        string backgroundColor;
        string textColor; string
        direction ;
        int speed;
};
```

Table 3-7 Variables

Variable	Description
text	Use this method to specify the text to be displayed.
fontSize	This method can be used to specify the font size.
fontFamily	Specify the font family using this method.
fontBold	Use this method if the specified text font is bold.
fontItalic	Use this method to display the specified text in italics.
backgroundImage	Use this method to include background image as a pure BASE64 data (without data: HTML prex).
backgroundColor	This method can be used to specify the background color.
textColor	Use this method to specify the text color in #RRGGBB format.
direction	Specify the scrolling direction. The possible values are: • LeftToRight • RightToLeft • TopToBottom • BottomToTop.
speed	Specify the scrolling speed. Possible values are from 1 to 12.

The following HTML document contains an example of ticker usage (file.../../test/plugins/scrollingtext.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <title>..:: scrollingtext plugin test ::.. </title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<s
tyl
e>
bo
dy
{
        margin: 0;
        background-color: #000000;
```

```
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
        color : #eeeeee;
}
</style>
<script type="text/javascript">

var
st ;
var
ind;
var ndirection = 1;

function init ()
{
        st = document.getElementById("scrollingtext"); ind = 0;
}
function start ()
{
        st . start () ;
}
function stop()
{
        st .stop() ;
}
function stext ()
{
        st . text = "New text #" + ind++;
}
function direction ()
{
        ndirection++;
        if (ndirection > 3)
                ndirection = 0;
        switch(ndirection)
        {
                case 0: st . direction = "LeftToRight"; break; case 1: st .
                direction = "RightToLeft"; break; case 2: st . direction =
                "TopToBottom"; break; case 3: st . direction =
                "BottomToTop"; break;
        }
}
function bold()
{
        st .fontBold = !st .fontBold;
}
function italic ()
{
        st . fontItalic = !st . fontItalic ;
}
</script>
</head>
<body onload="init()">
        <center>
                <br/>

                <div>
                <object id="scrollingtext" type="application/x-qt-plugin" classid="ticker" width="100%" height="80px">
                <!--
                All these parameters are available from JavaScript as object attributes . E.g.
                you can read and write them with
```

```
// get object
var st = document.getElementById("scrollingtext");
// set new text
st . text = "New text";
// set new font size st
. fontSize = 30;
// read current text
alert (st . text) ;
-->

<!-- text to show (empty by default) -->
<param name="text" value="Lorem ipsum dolor sit amet, consectetur adipiscingelit. Vestibulum ut sem leo." />
<param name="fontSize" value="24" /> <!-- font size (system-dependent by default)-->
<param name="fontFamily" value="Arial" /> <!-- font family (system-dependent by default) -->
<param name="fontBold" value="false" /> <!-- bold font (system-dependent by default) -->
<param name="fontItalic" value="false" /> <!-- italic font (system-dependent by default) -->

<!-- background image as a base64 data (empty by default) -->
<param name="backgroundImage" value="\
/9j/4AAQSkZJRgABAQEASABIAAD//gA8Q1JFQVRPUjogZ2QtanBlZyB2MS4wICh\
1c2luZyBJSkcgSlBFRyB2NjIpLCBxdWFsaXR5ID0gMTAwCv/bAEMAAQEBAQEBAQ\
EBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA\
QEBAQEBAQEBAf/bAEMBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB\
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAf/AABEIAPYAMAMBIgACEQE\
DEQH/xAAaAAADAQEBAQAAAAAAAAAAAAAAAAQQCAwUK/8QALxABAAIBAwMDAwIHAQ\
EBAAAAAREhMQJBUQASYQNxgRMikTKhQlJyscHR8GKS4v/EABsBAAIDAQEBAAAAA\
AAAAAAAAAECAAMEBQkK/8QALxEAAQIEBAUEAQQDAAAAAAAAAREhAAIxQVFhcfAD\
EoGRoROxwdHhIkJSYjLC8f/aAAwDAQACEQMRAD8A+ifXqNUQuFZw48FMzdQRQSZ\
XCVSBqcRkuLmtv9cter9s5sIL1IYIeW52v5kFol\
aIFwAVrm+MegQlJSyr4jt3ZBFtTOdqueYrFPFP1TGIWZdp2jFjAs9\
LXGkltXEwCzMTk5MzDVdLve0i0SkmhGq73rGu/GwbBaR905LsHJNsHT+pRGLLFo\
lWCZeO4iquR4/V0i/a8RUFcxMu8ruZnp/U0z+m3adJ/z4zmZlgqcTROkHkOW97v\
1PVY3L8/p8s5JtnfL031IiO6JZztNARwRBucnXL6xP6WY5N7D546PqXDo/EYvM+\
9uJkvoROQ4hb/ABaOOj1AZsgpYZOXKsbv7dadSzN/IHlEa2nLLWJ6h060iZ3kVW\
4iVuv2vJKB60MFaRhtVfBppaUObIvpZgVlIFD0sy0Bw+0XVLKrkB6AL7b0EVOtF\
BdTJgg2lmHE7TzjpfUXKjtZfARh9lpOpH1SXcim8kDmazny7vR9bMu2JbKfAYS4\
X7vfoiYMpGJpkuHkv3MWch/j7ZfQ0TKK31HZWhrTz8YzZOMX0PqJhU5gPzOI36k\
fUSftS/8A19xlSyRNkmM31l9RViYqbRUjzj2nxVg55cfBich/jcI2lcPhBhE5q1\
RHdxkC2c/atJ4JnjpOp5mPhBxO04AS5w1PF1Y+6ZPepG2Q+fLxKd7TOrnELUS25\
Xc3YsIgmkLMNQwHtbE2ZUjR6SIwNVQIVZHQZhnYRt1EqagZdmFmRZ2aOBt2Ot/U\
IzpmdmPZySNU4jaeoXVK0Ef1FITETupXA7da71rt0xEVMohYSGlvO3NdEygigbA\
AGyPfR0TCHHDKPXGtgoNFtS73i36k26K4XiIw/wDeK6f1UyDTlj/sUK7Rt1Gesb\
k7t2TaySXMREk+OuneWhmWMZZrcZf75CFDLKP5ai9N0HVoh4UwSinzolfESGuYZ\
tBPJExH4lGeKy+5tGAlLEnw+JkPHnqQ1gY1G0ycZ8xEmWZZ5O7eWNnbm7qXHLEx\
1SJ5TeN44ZAq7fC7+4ofUBYF4lGbCrkxDmyQgEwatTUunVO0LEFSKAfkxddcVbh\
kn4+CQzewcTPS79RH3cfzQFxB+ZlMUO9gmIAFQED1bOG9IGxVnS6BGS6Y6IkUuq\
9nxw7++rVw5h6Z6moWXuLkMudrgEjyhnqR7pzqJhgUNsGBMPlHEwd3lndllAuxf\
eJSMXXVksxmqKAfXxnC+kURAQRZQBTEDFVpnSJ31NQtsRLN9sMbXUxSLnFdM9TW\
R908GrTCtZx88TEu0/1GgdJmcIxMBOrVEPImd4OsuvUlm+0FryP7EqLEdc3mlYj\
mlOVHSoJFL0VOsaxlQHCqiMrnBtLnpHZ9UlrVBMo1OXYy75l9ug9TQ/xJnk5nk9\
pVziol1ayyGTJia03GMyxTUzwdyldr2mFhJDkPIcQ/F0syAIRMC9aKlbDowJKxa\
JWFQTlSgzYOigLdIrNemf1J+/wZl5ldqxDNRMSy6YqqMF2O8TJEsQxF3pA6Uc6h\
bSG7c/BFwY6RrZmCFSpIOcwmZ42Surgbg4UPV0v7GJyf2FU29con1a6QQmUhm/d\
pj7f1H5s6f1tYOlRH+7tLOahuuLibVrNQcilGDKaU+R97pnrKsx3OMZWJKmMWSo\
17dc9Ha6dygt89Uts5ZSjCu194pfVV0/ba47szJiqN44veF9TVOItbWHagTatv3\
Ym1a41WNFQSY3th5Te4ies6vV3043ki7/3NkLv1ZJKgVSMEREFPx4gjgiYDlDJU\
KQ4DF2SxGLg0is9UkUIyPcmMy/OI2nnrqa9ObLvOonmat8tDEQ9eeeoNRPiInHi\
IW7i811n6rMBczFDnETfn92OrASLpZV0RW8DzZZeDMHMoMtVIYeGQrnXBI4nqOr\
7TXZiInO0mXNZ28J16jVDqZX+JYY9wN/hfAdTHqQ+P4VqcmCvlZxt031R2YzD2x\
```

```
OTN3xNmKnrM60bFaU3rgI2+kDcMEpbJ/akVfVnVeYmS+SJYzEsThBqsuu9puSEk\
TZLV/Jk46k7+3UAJIEK8w4mZjzWzMpq9Q8ll6ri2J/8ALEWuH2bBOwAsnUBKJ+W\
xhpeEJURQSLFFC0x0XoLRW6pmZ0yREyXNkX3Ts5t2esd2o3YCPuj5JiiCF2cxfU\
+n1XLKMpTZHHETOeJnrL6gCprXu8wi4vOMzT+6mYl6d+mifcOP+7aOXfqJt1Rkb\
jb+WJuyyJurHXrLVqCQ2nFB5/J7kdqq4VI24tjjiuKU15HPliPMQ834V3jKJ5QQ\
5FLZjE9XuKIkaRIS/wCnIByihHpR+3Wg12ZdWSfu5Ga2p5siY6HVBkzRibXjMlX\
dSm8xJRt3bwapxG03/tm+k6naRsuUqWxNuXx4enHEEyULtQElkKG+jOcDE5Zrg1\
OHxaj+1BYanSfb21ZU+7msfKZs6HWrnTkgCcYxngHga2lfU1+JMyBGyb/au3D4j\
p/U1TB5RJ3TbtolxvBno88xsKljUIjVC/Lo0AyGqGzouCOQXtj1iTvilh2+1zN7\
Nw7EfM9Lv80Sz2sXN4eSaXOJeuD6m9lEMwIZ3xndiLt6X1C0PIwCzuXR5ZlozBz\
/AFCECLmvutc0jojghAiEoHIctcop7CKDUIjP4jjdPY9xYiOju0xluiIhZf2ov3\
6mdWmu0m7tj2L+6LmOKjpdyydumYl+6X8TnfY8nR9TLz+II4S2lfR6YB8NiKjVp\
unaMMhvXNYvzdhr0qS6uQvMzSyJE3O1QQdS96rQ247gnZf1Hx5hzTdWoMHdyfuw\
7tRmiJ5I4pYOKD/IsKeInp5S9sMGfavE2vWTptIf1csIm8bwEmYi4yOmhfiWPa2\
K35j46m16q7ndn7fEy/wrU3O8eesPqQNahC5R/wA0+d9i2KQVcbYb2sbpeCz+ai\
mB1Yq64taagntSvIGMF1+POemetDCEvkxfCGZ/y7dQ95Ew7Cu02JLWSxYZgz0n1\

QmmEhJM73ct3ghypHRhxw0YImpydFL9hSPQ+tpN9NVvn83/ANz1n6ir9zEu8AeN\
5mo6j+qP8LIl0Z2bfaL+HpPqFmEtXEpszWftoonk6kQcIGoBXVbXD4xPq1CELJq\
KmUElaxMYAhNr6z3u5K8wbMZ/EvtLvw1agDsRl2307HLLMZGLrGT1FG5vdJjzM1\
cfL56rlChQZhkCjtdMlGCo5eNg4YuJpq0YWCMCwQ37UipWP1UnGmLkynnm95VnJ\
rZtUf6cu0do0s1zJvHA9RxV8aZMTsQ1Y4/BIepq2InP22RWIJPJj8SQCFWYzWuM\
Kor5gBPMEcOVVImKuj5UcMUOissUOvUMTnglMqvbzUSn+Oh1kBYsWE3EpCZYmyY\
IXrhp16mViWSqxiyZPPiIYHpCk06p8lEQhLW8fEuejKoDub/Q3W8MOHUoh7YWiX\
X6ggGk0xqqGZp4CkmxceTrOnV2rFjzCu9m0Zm58Q9cPqKnmYmY2lXxAjx7z0TBk\
4oFkc7TMTLZJDnqkEgqGSNYkAD+KV33OUVHqJvzcBXGa8fFY6R6nl9oIjjN75yu\
NmFufOWP+8U0sv8AE9dA1BkmSSNm/L7hw9NzTM9NvBHCWi+PqLj1YAg+dX/5xxt\
iFk6f1v8Ayf8A1+/6cf4vHUU6pZdRs0Xds8c8SZ6zq1wx3Mh/KMHjEREO8xBwec\
3AhvS/qNWye+AxpHJ1GrfC8e2FjbiQMFdZXQBHa74zN1TzOaiOo31WR1Elxl/Vt\
IfksRVej6hBLI53Xgm94xzUyTnlsSTVqpXGjmNI4ZACBV0WrL42ClwGYPgI/wCM\
SZ+K33PL+XqM9WGki8i6cq8XFuZvFvR9XA6kpfPssJEHE+WGGQs+r4CXU1yuaKF\
ccKZKJ0dWr2/EVOtntG1mmIpvVEzzia33BdNsLM3JfwEu9/4nqQ1gqaoV+5mIn3\
JgrxMDOw+p3Qd2limdWmNxYqfebH46KpUsz4nR0B6X1gGQhgHerFNF7q7jOJJDV\
qIIoZXIywBlpzWLL66ZJvaN5Gvb7hs8T1x1T3M2S7XJW3xAzvZB0zVrKnbgj8Tn\
M81xaCZBKz65go16L2xA3ScOUOWFn0V8PkoopHVj+0tJjBMRmRiJfMdY1atsxM6\
iLiYPN5Ejz1z7ptbyz/ofMfjNPWe5Z7TGyP3fMWxGGvPQ5yAwpg9UxP4uihYYyp\
cDVFboWOD1xjqs/md/nff+/iulLsxc1zv8ss+/WGYlnONLz+6+J9q6HVFBLsabq\
MuKmPz7TVzTkuClsbI5KBaEeVhRw1/aOqfSxx+vMwMCUv8AUXHCU5KczJ9Weat8\
k4LrPyeb6OjpiAa+5jUJQwTD/T7MDrSWAuKzMLnhEHfeZyHrULLccRFO95bc7h0\
dHVQAKKtUqReUWOBMJC+qMzpV2vHjOOP+Og9UQY3HbEfmfnzM9HR0BJK1bfum/p\
nmepWJ37nL6j//2Q==\
" />

<param name="backgroundColor" value="" /> <!-- background color (white bydefault)
-->
<param name="textColor" value="" /> <!-- text color (black by default) -->
<param name="direction" value="RightToLeft" /> <!-- direction (RightToLeft by default, possible values:
LeftToRight, RightToLeft, TopToBottom,BottomToTop) -->
<param name="speed" value="7" /> <!-- speed [1..12] (8 by default) -->
</object>
</div>

<br/>

<a href='#' onclick="javascript:start()">Start</a>
<a href='#' onclick="javascript:stop()">Stop</a>
<a href='#' onclick="javascript:stext()">Change text</a>
<a href='#' onclick="javascript:direction()">Change direction</a>
<a href='#' onclick="javascript:bold()">Bold</a>
<a href='#' onclick="javascript: italic ()">Italic</a>
  </center>
</body>
</html>
```

VideoPlayer Widget

The VideoPlayer plugin allows you to play video in your application. It supports the majority of modern video and audio formats and codecs. It also allows you to use various transport protocols for your video (such as HTTP, RTSP, MMS, etc.).

This plugin also provides a minimal user interface to control video playback and volume. That user interface can be turned off, however.

The following declares the VideoPlayer JavaScript API:

```
interface VideoPlayer
 {

        attribute bool safeMode;
        attribute string url; attribute
        bool autoPlay; attribute bool
        loop; attribute int volume;
        attribute bool isMuted;
        attribute double speed;
        attribute double position;
        readonly attribute long long length; // Current media length in milliseconds. attribute long long time; // Current
        media time in milliseconds.
        readonly attribute bool isPlaying; readonly
        attribute bool isTrickPlaying; readonly attribute
        bool isPaused; readonly attribute bool isSeekable;
        attribute bool isFullScreen;
        readonly attribute int subtitlesCount; // Number of subtitle tracks. readonly attribute
        map<string, string> subtitlesMap;
        attribute int subtitle ; // Current subtitle track. Tracks numbers start from 1. readonly attribute string errorString;
        readonly attribute int audioTracksCount;
        readonly attribute map<string, string> audioTracksMap; attribute int
        audioTrack;

parameters:
        bool
        safeMode;
        bool
        autoPlay;
        bool   loop;
        int volume;
        bool muted;
        int fileCaching ; int
        networkCaching;
        bool hwAcceleration;
        string deinterlacing ; int
        fontSize ;
        string url ;

slots:

        int play(in string url); int
        pause();
        int
        resume();
        int stop();
        void setSubtitle (in int subtitle ) ;
```

```
        void fastForward (in double rate=10.0, in int frameDisplayTime = 500); void rewind (in double
        rate=10.0, in int frameDisplayTime = 500);

signals:

        void opening();
        void buffering();
        void started();
        void paused();
        void resumed();
        void stopped();
        void
        endReached();
        void error();

        void volumeChanged(in int volume); void
        muted(in bool muted);
        void seekableChanged(in bool seekable);
        void fullScreenMode(in bool enabled);
};
```

Table 3-8 Video Player Variables

Variable	Description
length	Current media length in milliseconds.
time	Current media time in milliseconds.
subtitlesCount	Number of subtitle tracks.
subtitlesMap	An associative array with subtitle's Id as a key, and subtitle's description as a value. **Caution** Remember that the Id is passed as a string. Use parseInt() to convert it to an integer. The array is not sorted by key.
subtitle	Current subtitle track.
audioTracksCount	Number of audio tracks.
audioTracksMap	An associative array with an audio track's Id as a key, and an audio track's description as its value. **Caution** Remember that the Id is passed as a string. Use parseInt() to convert it to an integer. The array is not sorted by key.
audioTrack	Current audio track.
isTrickPlaying	This method is set to True if fast forward or rewind is in progress.
safeMode	If true, the videoplayer object will recreate internal video object every time it starts playing.
autoPlay	Starts playing automatically.
loop	This is used to loop video playing.

volume	Specify the volume level, from 0 to 100.
muted	Starts in muted state.
fileCaching	File caching in milliseconds.
networkCaching	Network caching in milliseconds.
hwAcceleration	Specify whether you want to use hardware acceleration.
deinterlacing	Deinterlacing mode. If this method is set to auto, video player will try to deinterlace the interlaced video. • If it is turned on, deinterlacing is enforced. • If it is turned off, deinterlacing is off.
fontSize	Font size for subtitles.
url	URL to play.
fastForward	Starts fast forward playback. Rate defines the forwarding speed. For example, if rate == 10:0, it will forward ten times faster than the regular playback. • Rate must be greater than 1. • During fast forwarding, the player displays separate frames instead of contiguous video. frameDisplayTime defines the milliseconds each frame will be displayed. • frameDisplayTime must be greater or equal to 100; otherwise, 100 milliseconds will be used as its value.
rewind	Starts rewind playback. Rate defines the rewind speed. For example, if rate == 10:0, it will rewind ten times faster than the regular playback. • Rate must be greater than 1. • During rewind, the player displays separate frames instead of contiguous video. frameDisplayTime defines the milliseconds each frame will be displayed. • frameDisplayTime must be greater or equal to 100; otherwise, 100 milliseconds will be used as its value.

Note isFullScreen must be used before video playback is started; otherwise, it will only take effect after playback stop and restart.

Note You cannot use file:// URL in <param> tag of VideoPlayer object. Instead, you must set its url attribute at run time.

Note When fast forward or rewind is in progress, and player hits the movie end or beginning, the endReached signal is executed.

The following HTML document contains an example of VideoPlayer usage (file.../../test/plugins/videoplayer.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
        <title>..:: videoplayer widget example::.. </title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <s
    tyl
    e>
    bo
    dy
    {
        margin: 32px;
        background-color: #EEEEEE;
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
        color: #333333;
    }
    </style>
    <script type="text/javascript"> var
    player ;
    var subtitlesCount; var
    currentSubtitle ; var
    selectSubtitle ;
    var checkBoxFullScreen;
    var checkBoxLoop;
    var isVisible = true; var
    duration, position ;
    var endReached, endReachedCount = 0;

    function update()
    {
        if (player . isPlaying
                || player . isTrickPlaying)
        {
            duration.innerHTML = Math.round(player.length / 1000.0) + " seconds"; position .innerHTML =
            Math.round(player.time / 1000.0) + " seconds";

            // reset substitles . TODO: subtitles in the streamed media may appear after a small delay.
            // In this case we need to check for them all the time. In the case of a local file we probably
            // don't need that logic and can call updateTrackInfo() only once when the video has started
    updateTrackInfo();
        }
        window.setTimeout(update, 1000);
    }

    function updateTrackInfo()
    {
        var c = player. subtitle ;
        currentSubtitle .innerHTML = c >= 0 ? c : "-"; var n =
```

```
        player.subtitlesCount;
        if (n == selectSubtitle.options.length) return;
        subtitlesCount.innerHTML = n;
        // remove all elements
        for (var i = selectSubtitle
        .options.length
        •                                                                    •
        1; i >= 0; i--)
        {
        selectSubtitle .remove(i);
        }
        var map = [];
        // id => description mapping
        var apiMap = player.subtitlesMap;
        // the mapping is not sorted, let ' s sort it by id for (var i in apiMap)
        {
        map.push({id: parseInt(i) , description : apiMap[i]});
        }
        map.sort(function(a, b) { return a.id
        •                                                                    •
          b.id; } ) ;
        // append new elements
        for (var i in map)
        {
        var opt = document.createElement("option"); opt.value =
        map[i].id;
        opt.innerHTML = "" + map[i].id + ": " + map[i].description; selectSubtitle .appendChild(opt);
        }
}
function incEndReached()
{
        endReached.innerHTML = ++endReachedCount;
}

function main()
{
        subtitlesCount = document.getElementById("subtitlesCount"); currentSubtitle =
        document.getElementById("currentSubtitle"); selectSubtitle =
        document.getElementById("selectSubtitle"); checkBoxFullScreen =
        document.getElementById("checkBoxFullScreen"); checkBoxLoop =
        document.getElementById("checkBoxLoop"); checkBoxVisible =
        document.getElementById("checkBoxVisible"); checkBoxFullScreen.checked =
        player.isFullScreen; checkBoxLoop.checked = player.loop;
        duration = document.getElementById("duration"); position =
        document.getElementById("position"); update();
        window.setTimeout(update, 1000);
}

function error(message)
{
        alert ("Video player error : " + message);
}

function init ()
{
        player = document.getElementById("player"); if (player
        . status == 0) // 0 means ready. main();
        else
        {
        player .ready.connect(main); player .
        error .connect(error) ;
```

```
            }
            player . started .connect(updateTrackInfo); player
            .endReached.connect(incEndReached);
        }
        function setSubtitle ()
        {
            player . subtitle = selectSubtitle .value;
        }
        function setFullScreen ()
        {
            player . isFullScreen = checkBoxFullScreen.checked;
        }
        function setLoop()
        {
            player .loop = checkBoxLoop.checked;
        }
        function setVisible ()
        {
            isVisible = ! isVisible ;
            player . style . display = isVisible ? "block" : "none";

            function setSpeed(speed)
            {
            player .speed = speed;
            }
        }
    </script>
    </head>
<body onload="init()">
    <table width="80%" cellspacing="8" cellpadding="0" border="0" align="center" >
        <tr>
        <td width="50%" align="center">
        <object id="player" type="application/x-qt-plugin" classid="videoplayer" width="720" height="480" >
        <!-- <param name="url" value="http://html5videoformatconverter.com/data/images/happyfit2.ogv"/>
        -->
        <!-- <param name="url" value="mms://media.sonoma.edu/pinball.wmv"/>
        •                                                                                                    •
        -->
        <!-- <param name="url" value="file:///var/www/v/p.mp4"/>-->
        <param name="url" value="http://192.168.0.110/video/im_glad_cc.mov"/>
        <param name="volume" value="50"/> <!-- 75 by default -->
        <param name="autoPlay" value="true"/>
        </object>
        <br />
        <button onclick="player.play()">Play</button>
        <button onclick="player.stop()">Stop</button>
        <button onclick="player.pause()">Pause</button>
        <button onclick="player.fastForward()">Fast forward</button>
        <button onclick="player.rewind()">Rewind</button>
        </td>
        <td width="50%">
            <table width="100%" align="center" cellspacing="8" cellpadding="0" border="0" >
                <tr>
                    <td align="right">
                        Player is visible
                    </td>
                    <td>
                        <input id="checkBoxVisible" type="checkbox" checked="1"
        onchange="setVisible()" />
                    </td>
                </tr>
```

```
    <tr>
    <td align="right"> Subtitles
            available :
    </td>
    <td>
            <span id="subtitlesCount">0</span>
    </td>
    </tr>
    <tr>
    <td align="right">
            Current subtitle :
    </td>
    <td>
    <span id="currentSubtitle"></span>
</td>
</tr>

<tr>
<td align="right">
      Change
      subtitle:
</td>
<td>
      <select id="selectSubtitle"></select>
      <input type="button" value="Apply" onclick="setSubtitle()" />
</td>
</tr>
<tr>
<td align="right">
      Full screen
      mode
</td>
<td>
   <input id="checkBoxFullScreen" type="checkbox" onchange="setFullScreen()" />
</td>
</tr>
<tr>
<td align="right">
      Loop
</td>
<td>
      <input id="checkBoxLoop" type="checkbox" onchange="setLoop()" />
</td>
</tr>
<tr>
<td align="right">
      Duration:
</td>
<td>
      <span id="duration"></span>
</td>
</tr>
<tr>
<td align="right">
   Current position:
</td>
<td>
   <span id="position"></span>
</td>
</tr>
<tr>
<td align="right">
   Speed:
```

```
                </td>
                <td>
                    <a href='#' onclick="setSpeed(1)">1x</a>
                    <a href='#' onclick="setSpeed(2)">2x</a>
                    <a href='#' onclick="setSpeed(3)">3x</a>
                </td>
                </tr>
                <tr>
                <td align="right">
                    endReached signal ?red:
                </td>
                <td>
                    <span id="endReached"></span>
                </td>
                </tr>
                </table>
                </td>
                </tr>
                </table>
        </body>
    </html>
```

Copy the above text to an html file and transfer the file to a web server from where it can be accessed. Ensure to have the location of the file as a URL.

There are two ways to test the video player widget:

1. Using the policy:
    1. Create a policy with startup URL as the URL of the video player html and apply the policy
    2. Reboot the IEC

        As a result, the IEC boots up with the policy loaded.
2. Using the Kiosk menu:
    1. From IEC, press Ctl + Alt + S and then choose Kiosk
    2. Enter the video player widget URL
    3. Reboot the IEC

        As a result, the IEC boots up with the video player URL loaded.

VncViewer Widget

The VncViewer provides a VNC viewer in the form of a browser widget.

```
interface VncViewer
{
    readonly attribute int port; readonly
    attribute bool isConnected; attribute bool
    readOnly;
    attribute bool scalingEnabled;
    bool connect(in string host, in int port,in string password = "");

parameters:
    string
    host; int
```

```
        port;
        bool autostart ;
        bool scale ;
        bool readonly;
        string
        password;

slots :
        void disconnect() ;
signals :
        void connected();
        void disconnected();
        void error(in string message);
};
```

Table 3-9 VncViewer Variables

Variable	Description
port	Specifies port number to which the viewer is connecting. Default value is 5900.
readOnly	Specifies if the viewer accepts any user input.
connect(in string host, in int port, in string password)	Use this method to connect to a VNC server running at the host and listening port. **Note** This method accepts a port and not a display number.
isConnected	Returns 'true' if the viewer is connected to a VNC server.
host	The host to which the viewer is connecting.
autostart	Use this method to start the connection immediately.
Scale	Specify the scaling factor.
readonly	Use this method if the controls are disabled.
password	Specify the password if the connection is protected.
disconnect()	Closes the connection to a VNC server.
connected()	This method can be executed when a connection to a VNC server is established. This signal executes for both active and listening mode.
disconnected()	This method can be executed when the connection to a VNC server breaks.
error(in string message)	This method can be executed when an error occurs.

The following HTML document contains an example of VncViewer usage (file.../../test/plugins/vncviewer.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
	<title>..:: vncviewer plugin test ::..</title>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<s

tyl

e>

bo

dy
{
	margin: 0;
	background-color: #000000;
	font-weight: bold;
font-family: Arial; font-size: 20px; color: #eeeeee;


}

</style>

<script type="text/javascript"> var v;
var connectedId;

function init()
{
	v = document.getElementById("vncviewer"); connectedId =
	document.getElementById("connected");

	v.connected.connect(connected);
	v.disconnected.connect(disconnected);


	disconnected();
}


function connected()
{
	connectedId.innerHTML = "Connected";
}


function disconnected()
{
	onnectedId.innerHTML = "Disconnected";
}


function disconnect()
{
try
{

v.disconnect();

}
catch(ex)
{
alert("Exception: " + ex);
}
}
```

```
function connect()
{

try
{

v.connect("vnc://192.168.0.105");

// or
// v.connect("vnc://192.168.0.105", 5900);

// or
// v.connect("vnc://192.168.0.105", 5900, "my-password");

// or
/ v.connect("vnc://192.168.0.105", -1, "my-password");
}
catch(ex)
{
alert("Exception: " + ex);
}
}

function setReadOnly(ro)
{
Try
{
v.setReadOnly(ro);
}
        catch(ex)
        {
                alert("Exception: " + ex);
        }
}

function isReadOnly()
{
                        alert(v.readOnly);
                }

</script>

</head>

<body onload="init()">
        <center>
        <br/>

        <div>
        <object id="vncviewer" type="application/x-qt-plugin" classid="vncviewer" width="640px" height="480px">
                <!-- host to connect to -->
                <param name="host" value="vnc://192.168.0.105" />

                <!-- port to connect to. "5900" by default -->
                <param name="port" value="5900" />

                <!-- password to access the host (optional). Don't use this field if a server doesn't have a password -->
                <!-- <param name="password" value="123" /> -->

                <!-- fit to window? "true" by default -->
                <param name="scale" value="true" />

                <!-- ignore user input? "true" by default -->
```

```
        <param name="readonly" value="false" />

        <!-- start connection right now? If "false", no connection is established and all other parameters are ignored. If
        "true", start a VNC connection right now. "true" by default -->
        <param name="autostart" value="true" />
    </object>
    </div>

    <br/>

        <a href='#' onclick="javascript:isReadOnly()">Is read-only?</a>
        <a href='#' onclick="javascript:disconnect()">Disconnect</a>
        <a href='#' onclick="javascript:connect()">Connect</a>

    <div id="connected" style="text-align: center;">
    </div>
    </center>
    </body>
</html>
```

WebBrowser Widget

The WebBrowser plugin is an alternative of the iframe HTML element. In contrast to iframe, it allows to completely control the web document loaded within it and does not obey the same origin policy.

The WebBrowser interface declaration is the following:

```
interface WebBrowser
{

    attribute string url;
    attribute bool navigationPanelVisible; attribute
    bool titlePanelVisible; attribute double
    zoomfactor;

parameters:
    string url ;
    bool navigationPanelVisible; bool
    titlePanelVisible ; double
    zoomFactor;

slots:
    variant evaluateJavaScript(in string scriptSource); signals:

    void loadStarted();
    void loadFinished(in bool ok); void
    loadProgress(in int percent); void
    urlChanged(in string url); void linkClicked(in
    string url); void titleChanged(in string title);

};
```

Table 3-10 Variables

Variable	Description
url	URL to display.
navigationPanelVisible	Use this method to make the navigation panel visible.
titlePanelVisible	Use this method to display the title.

zoomFactor	Specify the zoom factor. It is set to 1.0, be default.

The following HTML document contains an example of WebBrowser usage (file../../test/plugins/webbrowser.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

    <head>

        <title>..:: webbrowser plugin test ::..</title>

        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

     <style

     >

     body
     {
          margin: 0;
          background-color: #000000;
          font-weight: bold;
          font-family: Arial;
          font-size: 20px;
          color: #eeeeee;
     }

     </style>

     <script type="text/javascript"> var

    addressBar;
    var webBrowser;

     function init()
     {
          addressBar        =        document.getElementById("addressBar");
          webBrowser        =      document.getElementById("webBrowser");
          addressBar.value = webBrowser.url;
     }
    }
    function go()
    {
    webBrowser.url = addressBar.value;

     </script>
     </head>

     <body onload="init()">

          <input id="addressBar" type="text" />
          <button onclick="go()">Go</button>
          <br />
          <br />
          <font
          color="#FFFFFF">WebKit
          •                                                                                    •
          based browser</font>
          <div style="text-align: center ;">
```

```
        <object id="webBrowser" type="application/x-qt-plugin" classid="webbrowser" width="800" height="600">
        <param name="url" value="http://google.com"/>
        <param name="navigationPanelVisible" value="false"/>
        <param name="titlePanelVisible" value="false"/>
        <param name="zoomFactor" value="2.0"/>
        </object>
        </div>
        </body>
</html>
```

WebClip Widget

The WebClip plugin is inspired by Mac OS X dashboard plugin with the same name. This plugin allows you to include other web site elements in your web application. You need the web site URL and the CSS selector that addresses the element that you want to include.

```
interface WebClip
 {
      readonly attribute string url; readonly
      attribute string selector;
      attribute int reloadInterval; // In seconds.

      void load();
 };
```

The following HTML document contains an example of WebClip usage (file.../../test/plugins/webclip.html). This page contains two webclips. They take elements from Yahoo! Finance and Yahoo! Weather web sites.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
      <title>..:: webclip widget example ::..</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<s
tyl
e>
bo
dy
{
      margin: 32px;
      background-color: #EEEEEE;
      font-weight: bold;
      font-family: Arial;
      font-size: 20px;
      color : #333333;
}
</style>
<script type="text/javascript"> var
stockClip;
var
weatherClip;
function init ()
{
      stockClip = document.getElementById("stockClip"); weatherClip =
      document.getElementById("weatherClip");
}
</script>
</head>
<body onload="init()">
      <table border="0" cellpadding="20" cellspacing="20" align="center" width="90%">
      <tr>
```

```
    <td align="center">
        <object id="stockClip" type="application/x-qt-plugin" classid="webclip" width="320" height="240">
        <param name="url" value="http://finance.yahoo.com/q?s=CSCO&ql=1"/>
        <param name="selector" value=".chart"/>
        <param name="reloadInterval" value="600"/>
        <param name="timeout" value="5"/>
        <param name="progressIndicatorVisible" value="false"/>
        </object>
    <br />
  <input type="button" value="Reload" onclick="stockClip.reload();" />
    </td>
    </tr>
    <tr>
    <td align="center">
        <object id="weatherClip" type="application/x-qt-plugin" classid="webclip" width="950" height="280">
        <param name="url" value="http://www.cisco.com"/>
        <param name="selector" value="#spotlight-container"/>
        <param name="reloadInterval" value="300"/>
        </object>
    <br />
    <input type="button" value="Reload" onclick="weatherClip.reload();" />
    </td>
    </tr>
    </table>
</body>
</html>
```

WebRTC Widget

WebRTC (Web Realtime Communications) enables peer to peer video, audio, and data communication in Cobra. This allows video calling, video chat, and peer to peer file sharing entirely in Cobra browser, with no plugins. Additionally, it can be used for screen sharing, presentation sharing, and so on. It can work with any WebRTC service provider such as Google AppSpot, opentokrtc.com, easyrtc.com, and so on.

The interface declaration is the following:

```
interface rtcclient
{

    attribute string url ;
methods:

    int startClient (in string url) ;

    void runJavaScript(in string scriptSource) ;

    variant evaluateJavaScript(in string scriptSource) ; int stopClient() ;

parameters:
    string url ;

signals :
    void loadStarted();
    void loadFinished(in bool ok); void
    loadProgress(in int percent); void
    urlChanged(in url url) ;
    void titleChanged(in string title ) ; void error(in
    string errorMsg);

};
```

Table 3-11 Variables

Variable	Description
startClient	Use this method to initialize the widget with the desired URL.
runJavaScript	Executes the specified javascript code *ScriptSource* in the RTC client. The javascript code is executed asynchronously.
evaluateJavaScript	Executes the specified javascript code *ScriptSource* in the RTC client. The javascript code is executed synchronously and the result is returned.
stopClient	Use this method to deinitialize the widget.
url	URL pointing to the HTML document.

The following HTML document contains an example of RTC client usage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
                <title>..:: webrtc plugin test ::.. </title>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <s

    tyl

    e>

    bo

    dy
    {
        margin: 0;
        background-color: #000000;
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
        color : #eeeeee;
    }

    </style>

    <script type="text/javascript" src="rtcclient . js"></script>

    <script type="text/javascript"> var

    addressBar;
    var rtcclient ;

    function init ()
    {
        addressBar = document.getElementById("addressBar"); rtcclient =
        document.getElementById("rtcclient"); addressBar.value = rtcclient
        . url ;
        setTimeout(start, 3000); // Call start after 3 seconds
    }

    function deinit ()
    {
```

```
        rtcclient .stopClient() ;
}

function start ()


{
rtcclient . startClient (addressBar.value);
}

</script>

</head>
<body onload="init()" onbeforeunload="deinit()">
<input id="addressBar" type="text" align="center"/>
<button onclick="start()" align="center">Start</button>
<br />
<br />
<font color="#FFFFFF"><center>..:: webrtc plugin test ::..</center></font>
<div style="text-align:center ;width:1280px;height:720px;margin:0 auto;">
        <object id="rtcclient"
                type="application/x-qt-plugin" classid="rtcclient"
                style="width:100%;height:100%">
                <param name="url" value="https://apprtc.appspot.com/"/>
        </object>
</div>
</body>
</html>
```

JavaScript Global Objects

Chapter Overview

In addition to standard JavaScript global objects (such as window or document) COBRA provides several non-standard global objects to allow web applications to be better integrated with the IEC. Those objects implement functionality which is not available in a regular browser's JavaScript.

The topics in this chapter include:

global.applicationData Object

The global.applicationData object allows the web application to store its persistent data on the management server. This object is a convenience API to access application.data property. Once you change the data, it will automatically be saved to the management server.

This object does not allow you to control the application in real time. For example, if you construct some policy containing application.data property on the management server and apply that policy as an action to the device running your application, the application will not know about that until the next reboot. Moreover, if the application changes the data after that policy has been applied, the changes you are trying to make with that policy will be lost. This approach tries to prevent inconsistent behavior of the application.

```
interface ApplicationData
{
        readonly attribute bool isReadOnly; readonly
        attribute  int  size; readonly attribute bool
        isEmpty; readonly attribute List<String> keys;
        readonly attribute List<String> values;

        bool contains(in String key) const; String
        value(in String key) const;
        String value(in String key, in String defaultValue) const; void insert(in String key,
        in String value);
        bool remove(in String key);
        String take(in String key); // Returns value and removes it from the data. slots:

        void clear();

};
```

Sometimes the value of application.data property can be defined with a policy. That means it cannot be changed on the management server from the device side. In that case isReadOnly attribute returns true. The application still can change its data using insert(), remove(), take() or clear() methods. However, the data will not be saved on the management server.

global.bus Object

The global.bus is an unidirectional bus to communicate with other Cobra instances. This object implements a Bus interface:

```
interface Bus
{
      void send(in object obj) const; bool
      haveOtherInstances() const;
signals :
      void received(in object obj)
};
```

Table 4-1 global.bus variables

Variable	Description

send()	Sends the given object to other Cobra instances. The object itself can represent simple types (e.g. integers, strings, etc.) as well as complex types like associative arrays.
haveOtherInstances()	Returns true if there are other Cobra instances that can receive events.
received()	Fires when an event has arrived from another Cobra instance.

The following HTML is an example of global.bus usage for sending events (file.../../test/js/bus-send.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
    <title>..:: global .bus test ::.. </title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <s
    tyl
    e>
    bo
    dy
    {
        margin: 20px;
        background-color: #111111;
        color: #eeeeee;
        font-weight: bold;
        font-family: Arial;
        font-size: 18px;
        color: #eeeeee;
    }
    </style>
    <script type="text/javascript"> var
    instancesId ;
    var textId;
    function init ()
    {
        instancesId = document.getElementById("instances"); textId =
        document.getElementById("text");
    }
    function enumerate()
    {
        try
        {
        var have = global.bus.haveOtherInstances(); if (!have)
        instancesId .innerHTML = "<font color='#?0000'>NOT DETECTED</font>"; else
        instancesId .innerHTML = "<font color='#00?00'>DETECTED</font>";
        }
        catch(ex)
        {
        alert ("Exception: " + ex);
        }
    }
    function send()
    {
        try
        {
        global .bus.send({id: textId.value});
        }
        catch(ex)
        {
        alert ("Exception: " + ex);
        }
```

```
    }
    </script>
    </head>
    <body onload="init()">
        <input type="button" value="Detect other instances" onclick="enumerate()" /><span id="instances"></span><br>
        <input id="text" type="text" /><input type="button" value="Send as { id: <value>
        }" onclick="send()" />
    </body>
</html>
```

The following HTML is an example of global.bus usage for receiving events (file.../../test/js/bus-receive.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
    <title>..:: global .bus test ::.. </title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <s
    tyl
    e>
    bo
    dy
    {
        margin: 20px;
        background-color: #111111;
        color: #eeeeee;
        font-weight: bold;
        font-family: Arial;
        font-size: 18px;
        color: #eeeeee;
    }
    </style>
    <script type="text/javascript">
    function init ()
    {
        global .bus.received .connect(recieve) ;
    }
    function deinit ()
    {
        global .bus.received .disconnect(recieve ) ;
    }
    function recieve (value)
    {
        alert ("Received data: " + value + "\nID: " + value.id);
    }
    </script>
    </head>
    <body  onload="init()"  onunload="deinit()"> Waiting for events
        with { id : &lt;value&gt; }
    </body>
</html>
```

global.database Object

The global.database object can open or remove a database and list its files.

```
interface Database
{
    object open(in string uuid, int string name, int int version , int string sql ) ; bool remove(in string
    uuid, in string name);
```

```
    list <string> list(in string uuid);
};
```

Table 4-2 global.database Variables

Variable	Description
open()	Opens a database with name *name* and version *version*. If the database version doesn't match the version passed as an argument then it is recreated using SQL sql. sql must contain a single SQL statement per line. Every SQL statement must be followed one empty line. Returns a database object on success. You also need to check the database status (see below). The database itself is based on SQLite. Refer to SQLite documentation on how to use its SQL statements.
remove()	Removes the given database name from the local storage.
list()	Lists the database files available in the local storage.

The open() variable returns an object, which implements the DatabaseObject interface.

```
interface DatabaseObject
{
    readonly attribute int status ;
    readonly attribute list <list<value>> result; void close () ;

    bool   begin();
    bool commit();
    bool rollback () ;

    bool exec(in string sql , in map<string, variant> binds = map<string, variant>());
}
```

Table 4-3 DatabaseObject Variables

Variable	Description
status	The database status. 0=unknown, 1=healthy, 2=migrated, 3=recreated, 4=created.
result	The result of the last call to exec().
close()	Closes the database discarding any uncommitted data.
begin()	Starts a new transaction.
commit()	Commits the transaction.
rollback()	Rollbacks the transaction.
exec()	Executes a single SQL statement sql and stores its result in result.

The following HTML document contains an example of global.database usage.

```
<!DOCTYPE HTML PUBLIC "//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
	<title>..:: global .database example ::..</title>
	<meta httpequiv="ContentType" content="text/html; charset=utf8">
<s
tyl
e>
bo
dy
{
	margin: 32px;
	backgroundcolor: #EEEEEE;
	fontweight: bold; fontfamily:
	Arial; fontsize: 20px;
	color : #333333;
}
</style>
<script type="text/javascript"> var
log;
var number;
var
explanation;
var db;
// unique application id in any format (UUID v1/v4 is recommended) var UUID =
"69c6fb59b4bf4afa9a282bee7eb1ca10";
function init ()
{
	log = document.getElementById("log"); number =
	document.getElementById("number");
	explanation = document.getElementById("explanation");
	/*
	SQL string must contain just a single SQL statement per line. Every SQL statement must be followed by
	one empty line (e.g. by two \n characters).
	*/
	db = global.database.open(UUID, "numbers", 2, "\ DROP TABLE
	IF EXISTS numbers; \n\n\
	VACUUM; \n\n\
	CREATE TABLE numbers \
(\
number INTEGER, \
explanation VARCHAR(128) \
)
;
\
"
)
;
	log .innerHTML += "DB status: " + db.status + "<br><br>Explanation:<br>0 = unknown<br>1 = healthy<br>2
	= migrated<br>3 = recreated<br>4 = created<br>"; log .
	scrollTop = log. scrollHeight ;
}
function deinit ()
{
	db.close () ;
}
function list ()
{
	alert (global .database. list (UUID));
}
function cleartbl ()
{
	log .innerHTML += "CLEAR: "
	+ (
```

```
        db.exec("DELETE FROM numbers")
        ? "OK" : "FAILED"
        )
        + "<br>";
        log . scrollTop = log. scrollHeight ;
}
function unlink()
{
        alert ("Removed: " + (global.database.remove(UUID, "numbers") ? "yes" : "no"));
}
function insert ()
{
        log .innerHTML += "INSERT ‘"
        + number.value
        + "': "
        + (
        db.exec("INSERT INTO numbers VALUES (" + number.value + ", '" + explanation.value
        +
        "
        '
        )
        "
        )
        ? "OK" : "FAILED"
        )
        + "<br>";
        /*
        Another way to do exec():
        var binds = [];
        binds[":num"] = number.value; binds[":explanation"] =
        explanation.value;
        db.exec("INSERT INTO numbers VALUES (:num, :explanation)", binds);
        */
        log . scrollTop = log. scrollHeight ;
}
function insertTransaction()
{
if (!db.begin())
{
log .innerHTML += "Cannot start transaction<br>"; log . scrollTop =
log. scrollHeight ;
return;
}
var tobe = 5;
var actual = 0;
var ret ;
for (var i = 0;i < tobe;i++)
{
        log .innerHTML += "INSERT ‘"
        + number.value
        + "': "
        + (
        (ret = db.exec("INSERT INTO numbers VALUES (" + number.value + ", '" + explanation.value + "')"))
        ? "OK" : "FAILED"
)
+ "<br>";

actual += ret;
}
if (actual != tobe)
        {
```

```
                db.rollback() ;
                log .innerHTML += "Cannot exec some statements<br>"; log . scrollTop
                = log. scrollHeight ;
                return;
                }
                if (!db.commit())
                {
                log .innerHTML += "Cannot commit transaction<br>"; log .
                scrollTop = log. scrollHeight ;
                return;
                }
                log . scrollTop = log. scrollHeight ;
        }
        function select ()
        {
        db.exec("SELECT * FROM numbers");
        log .innerHTML += "SELECT [size:" + db.result.length + "]: " + db.result + "<br>"; log . scrollTop = log. scrollHeight ;
        }
        </script>
        </head>
        <body onload="init()" onunload="deinit()">
                <div style="textalign: center ;">
                <button onclick="list()">List databases</button>
                <input id="number" type="number" value="0" style="width: 60px">
                <input id="explanation" type="text" placeholder="explanation" style="width: 100px">
                <button onclick="insert()">Insert</button>
                <button onclick="insertTransaction()">Insert with a transaction</button>
                <button onclick="select()">Select all</button>
                <button onclick="cleartbl()">Clear table</button>
                <button onclick="unlink()">Remove database 'numbers'</button>
                <br>
                <span id="log" style="height: 300px; display: block; overflow: auto;"></span>
                </div>
        </body>
</html>
```

global.device Object

It is possible to obtain some general device-related information using global.device global object. This object implements device interface.

```
interface Device
{
        readonly attribute string name; readonly
        attribute string description; readonly attribute
        string product; readonly attribute string family;
        readonly attribute string model; readonly attribute
        string version; readonly attribute string
        versionSystem;
        readonly attribute string versionApplications; readonly
        attribute string location;
        readonly attribute string serialNumber; readonly
        attribute string licenseKey; readonly attribute bool
        licenseValid; readonly attribute string
        buildTimestamp;
        readonly attribute string buildTimestampSystem; readonly attribute
        string buildTimestampApplications;
};
```

Table 4-4 global.device Variables

Variable	Description
name	device name
description	device description that can contain any text supplied by administrator
product	product name
family	family name that specifies device hardware in general so you can use model name to make assumptions about device capabilities
model	model name that specifies device hardware (i.e. how much memory it has)
version	device firmware version
versionSystem	device system firmware version
versionApplications	device applications firmware version
location	device location that contains text supplied by administrator
serialNumber	device serial number
licenseKey	device license key
licenseValid	returns 'true' if the license key is valid or 'false' if otherwise
buildTimestamp	firmware build time stamp in human-readable form
buildTimestampSystem	system firmware build time stamp in human-readable form
buildTimestampApplications	applications firmware build time stamp in human-readable form

The following is the global.device usage in HTML (file../../test/js/device.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
 <html>
 <head>
      <title>..:: global.device test ::..</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<style>

body
{
      margin: 20px;
      background-color: #000000;
      color: #eeeeee;
      font-weight: bold;
      font-family: Arial;
      font-size: 20px;
}


 </style>

 <script type="text/javascript">

 function init()
 {
      document.getElementById("name").innerHTML = global.device.name;
      document.getElementById("description").innerHTML = global.device.description;
      document.getElementById("product").innerHTML = global.device.product;
      document.getElementById("model").innerHTML = global.device.model; document.getElementById("version").innerHTML
      = global.device.version;

      document.getElementById("versionSystem").innerHTML = global.device.versionSystem;
```

```
        document.getElementById("versionApplications").innerHTML = global.device.versionApplications;
        document.getElementById("location").innerHTML = global.device.location;
        document.getElementById("serialNumber").innerHTML = global.device.serialNumber;
        document.getElementById("buildTimestamp").innerHTML = global.device.buildTimestamp;
        document.getElementById("buildTimestampSystem").innerHTML = global.device.buildTimestampSystem;
        document.getElementById("buildTimestampApplications").innerHTML = global.device.buildTimestampApplications;
}
  </script>

  </head>

   <body onload="init()">

        <table cellpadding="2" cellspacing="0" border="1px" align="center" width="50%">
              <tr>
              <td>Device name</td>
              <td id="name"></td>
              </tr>
              <tr>
              <td>Device description</td>
              <td id="description"></td>
             </tr>
              <tr>
              <td>Product</td>
              <td id="product"></td>
              </tr>
              <tr>
              <td>Model</td>
              <td id="model"></td>
              </tr>
              <tr>
              <td>Version</td>
              <td id="version"></td>
              </tr>
              <tr>
              <td>Version of system</td>
              <td id="versionSystem"></td>
              </tr>
              <tr>
              <td>Version of applications</td>
              <td id="versionApplications"></td>
              </tr>
              <tr>
              <td>Device location</td>
              <td id="location"></td>
              </tr>
              <tr>
              <td>Device serial number</td>
              <td id="serialNumber"></td>
              </tr>
              <tr>
              <td>Firmware build timestamp</td>
              <td id="buildTimestamp"></td>
              </tr>
              <tr>
              <td>System firmware build timestamp</td>
              <td id="buildTimestampSystem"></td>
              </tr>
              <tr>
              <td>Applications firmware build timestamp</td>
              <td id="buildTimestampApplications"></td>
              </tr>
```

```
        </table>

    </body>

</html>
```

global.display Object

This object implements the display interface.

```
interface System
{
    readonly attribute bool isBrightnessEnabled; attribute int
    brightness;

    readonly attribute bool isContrastEnabled; attribute
    int contrast;

    attribute int rotationAngle;
};
```

Note This object controls brightness and contrast using DDC (Display Data Channel). Not all displays support that so for some displays, brightness and contrast adjustment will not work.

Note This object in its implementation changes values of display.* properties. So, if some policy containing display.* properties is applied to the device, this object will not work.

Cali If you have multiple displays connected to the device, the rotationAngle attribute will rotate all the screens. As to brightness and contrast, in the current version only the first found display will be affected.

Table 4-5 global.display Variables

Variable	Description
isBrightnessEnabled	Returns value of display.brightness.enabled property (which is false by default). If it is false, display brightness cannot be adjusted.
isContrastEnabled	Returns value of display.contrast.enabled property (which is false by default). If it is false, display contrast cannot be adjusted.
rotationAngle	Gets and sets screen rotation angle. Only the following values are valid: 0, 90, 180 and 270. If you use an invalid value, this method does nothing.

The following HTML contains an example of global.display usage:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
        <title>..:: global.display test ::..</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<style> body
{
        margin: 20px;
        background-color: #000000;
        color: #eeeeee;
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
}

</style>

<script type="text/javascript"> var brightness;
var contrast;
var rotationAngle;

function init()
{
        brightness = document.getElementById("brightness"); contrast =
        document.getElementById("contrast"); rotationAngle =
        document.getElementById("rotationAngle");
        document.getElementById("minBrightness").innerHTML = global.display.minBrightness;
        document.getElementById("maxBrightness").innerHTML = global.display.maxBrightness;
        document.getElementById("isBrightnessEnabled").innerHTML = global.display.isBrightnessEnabled ?"Yes" : "No";
        brightness.value = global.display.brightness; document.getElementById("minContrast").innerHTML =
        global.display.minContrast; document.getElementById("maxContrast").innerHTML = global.display.maxContrast;
        document.getElementById("isContrastEnabled").innerHTML = global.display.isContrastEnabled ? "Yes" : "No";
        contrast.value = global.display.contrast; rotationAngle.value =
        global.display.rotationAngle;
}

function getBrightness()
{
                brightness.value = global.display.brightness;
}

function setBrightness()
{
        global.display.brightness = brightness.value;
}

function getContrast()
{
        contrast.value = global.display.contrast;
}

function setContrast()
{
        global.display.contrast = contrast.value;
}
function getRotationAngle()
{
        rotationAngle.value = global.display.rotationAngle;
}

function setRotationAngle()
{
```

```
        global.display.rotationAngle = rotationAngle.value;
}

</script>

</head>

 <body onload="init()">

        <table cellpadding="2" cellspacing="0" border="1px" align="center" width="50%">
                <tr>
                <td>Minimum brightness</td>
                <td id="minBrightness"></td>
                </tr>
                <tr>
                <td>Maximum brightness</td>
                <td id="maxBrightness"></td>
                </tr>
                <tr>
                <td>Brightness adjustment enabled</td>
                <td id="isBrightnessEnabled"></td>
                </tr>
                <tr>
                <td>Current brightness</td>
                <td>
                <input id="brightness" type="text" size="3" maxlength="3" />
                <input type="button" value="Get" onclick="getBrightness()" />
                <input type="button" value="Set" onclick="setBrightness()" />
                </td>
                </tr>
                <tr>
                <td>Minimum contrast</td>
                <td id="minContrast"></td>
                </tr>
                <tr>
                <td>Maximum contrast</td>
                <td id="maxContrast"></td>
                </tr>
                <tr>
                <td>Contrast adjustment enabled</td>
                <td id="isContrastEnabled"></td>
                </tr>
                <tr>
                <td>Current contrast</td>
                <td>
                <input id="contrast" type="text" size="3" maxlength="3" />
                <input type="button" value="Get" onclick="getContrast()" />
                <input type="button" value="Set" onclick="setContrast()" />
                </td>
                </tr>
                <tr>
                <td>Display rotation angle (can be 0, 90, 180 or 270)</td>
                <td>
                <input id="rotationAngle" type="text" size="3" maxlength="3" />
                <input type="button" value="Get" onclick="getRotationAngle()" />
                <input type="button" value="Set" onclick="setRotationAngle()" />
                </td>
                </tr>
        </table>

</body>
```

</html>

global.ir Object

An Infrared (IR) Cisco Remote Control can be connected to the Moderro Interactive Experience Client 4600 (IEC 4600) Series device so that the end user can control applications and remote playback without touching the screen or using a mouse. Refer to the *Moderro Interactive Experience Client 4600 Series User Guide* for a list of supported remote controls.

The IR port is active by default. No additional configuration is required. The global.ir object implements the IR interface. It allows an application to receive signals from the infrared remote control.

Embed the global.ir object into your application code so that your applications will perform the expected action when the end user presses a button on the remote control.

The global.ir object code interface is the following:

```
interface Ir
{
        readonly attribute string lastError; List <String>
        availableControls() const;
        bool setCurrentControl(in string device); signals:
        event(in uint key, in string skey, in string configName) const; error(in string err) const;
}
```

Table 4-6 global.ir Object Variables

Variable	Description
lastError	Last error that occurred
availableControls()	Returns the list of the supported remote controls
setCurrentControl(in string device)	Sets the current remote control to use. The device name must be obtained from the availableControls() list. Leave the device name empty to use browser.ir.configuration. In this case, you should set browser.ir.configuration to the valid LIRC configuration.
event(in uint key, in string skey, in string configName)	Remote control event where the event control code is set to 'key', the control name (such as 'poweroff', 'ch1', 'up', etc.) is set to 'skey', and the configuration name, which is rarely needed, is set to 'configName'.
error(in string err)	An error has occurred. Use the method lastError() or argument 'err' to get the error string.

The following HTML code contains an example of global.ir usage (file../../test/js/ir.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>

    <head>

    <title>..:: global.ir test ::..</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <s

    tyl

    e>

    bo

    dy
    {
        margin: 20px;
        background-color: #111111;
        color: #eeeeee;
        font-weight: bold;
        font-family: Arial;
        font-size: 18px;
        color: #eeeeee;
    }

    </style>

    <script type="text/javascript"> var

    errorId;
    var eventId;
    var
    controlId;
    var timer;
        var remoteInfo;

    function init()
    {
        errorId = document.getElementById("error"); eventId =
        document.getElementById("event"); controlId =
        document.getElementById("control");
        remoteInfo = document.getElementById("remoteInfo");

        global.ir.error.connect(onError); writeLog("onError() connected to signal global.ir.error");
        global.ir.event.connect(onEvent); writeLog("onEvent() connected to signal global.ir.event");

            try{

                var irconf = global.registry.value("browser.ir.configuration");


                if(irconf=='<binary mimeType="text/plain"><![CDATA[]]></binary>' || irconf==''){
                        // set default remote as current.
                    //global.ir.setCurrentControl(defaultRemote); remoteInfo.innerHTML =
                    "Default (Cisco remote control)"; writeLog("Default remote control
                    configuration applied.");
                } else{
                        global.ir.setCurrentControl();
                    remoteInfo.innerHTML = "* user defined remote control *"; writeLog("User defined remote
                    control configuration applied.");
                }
            } catch(ex){
                writeLog("Exception: "+ex);
```

```
                }

        }

        function deinit()
        {
                global.ir.error.disconnect(onError);
                global.ir.event.disconnect(onEvent);
        }

        function onError(err)
        {
                errorId.innerHTML = err;
                        writeLog("onError(): "+err);
        }

        function onEvent(key, skey, config)
        {
                eventId.innerHTML = "" + key + ' ' + skey + ' ' + config; writeLog("onEvent():
                        "+key + " " + skey + " " + config);


                clearTimeout(timer);
                timer = setTimeout(function() { eventId.innerHTML = ""; }, 750);
        }


        </script>

        </head>

        <body onload="init()" onunload="deinit()"> Remote: <span
                id="remoteInfo"></span><br><br> Event: <span
                id="event"></span><br><br>
                Error log: <span id="error"></span>


        <!-- This part of code   is used for tracing application states -->
<div id="appDebugInfo" style="background:rgba(0,0,0,.9); position:absolute; top:0px; right:0; width:15%; min-width:300px;
bottom:0; overflow-x:hidden; border:solid 1px
#666666; overflow-y:scroll; color:white; padding:20px; font:normal 10px/12px sans-serif"></div>
<script>
function writeLog(msg){
        var c = document.getElementById('appDebugInfo'); if(c){
                        c.innerHTML=c.innerHTML + msg + "<br><br>";
        }
}
</script>
        </body>

</html>
```

global.keyboard Object

This object allows you to control a screen's keyboard. This keyboard can interfere with the popup screen keyboard. If you want to control screen keyboard in your JavaScript application, it is strongly recommended to turn the popup screen keyboard off by changing the setting browser.input.popup.keyboard.enabled property to false. You can use only one screen keyboard in your application.

```
interface Keyboard
{
        readonly attribute bool isVisible;

        void show(in int globalX, in int globalY); void hide();
        void move(in int globalX, in int globalY);
}
```

Note The globalX and globalY coordinates are relative to the DOM window.

The following is an example of global.keyboard usage in HTML (file ../../test/js/keyboard.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose. dtd">
<html>

<head>

    <title>..:: global.keyboard example ::..</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<s

tyl

e>

bo

dy
{
    margin: 32px;
    background-color: #EEEEEE;
    font-weight: bold;
    font-family: Arial;
    font-size: 20px;
    color: #333333;

}
</style>

<script type="text/javascript"> var
radioAutoKeyboardEnabled; var
radioAppKeyboardEnabled; var
keyboardPosition;

function showKeyboard()
{
    if (!radioAppKeyboardEnabled.checked) return;
    global .keyboard.show(300, 300);
}
function hideKeyboard()
{
    if (!radioAppKeyboardEnabled.checked) return;
    global .keyboard.hide();
}
function toggleAutoScreenKeyboard()
{
    global . registry .setValue("browser.screen.keyboard.enabled", radioAutoKeyboardEnabled.checked);
}
function setAutoScreenKeyboardPosition()
```

```
{
        global . registry .setValue("browser.screen.keyboard.position", keyboardPosition.value);
}
function init ()
{
        var text = document.getElementById("text");
        radioAutoKeyboardEnabled = document.getElementById("radioAutoKeyboardEnabled"); radioAppKeyboardEnabled =
        document.getElementById("radioAppKeyboardEnabled"); keyboardPosition =
        document.getElementById("keyboardPosition");
        text   .onfocus   =   showKeyboard; text .
        onclick = showKeyboard; text .onblur =
        hideKeyboard;
        radioAutoKeyboardEnabled.checked =
        global.registry.value("browser.screen.keyboard.enabled") == "true"; radioAppKeyboardEnabled.checked =
        !radioAutoKeyboardEnabled.checked; radioAutoKeyboardEnabled.onclick = toggleAutoScreenKeyboard;
        radioAppKeyboardEnabled.onclick = toggleAutoScreenKeyboard;
        keyboardPosition.value = global. registry .value("browser.screen.keyboard.position"); keyboardPosition.onchange =
        setAutoScreenKeyboardPosition;
}
</script>
</head>
 <body onload="init()">
        <div style="text-align: center ;"> Text
                input
                <input id="text" type="text" />
                <br />
                <br />
                <input id="radioAutoKeyboardEnabled" type="radio" name="keyboardType" /> Browser automatic screen
                keyboard enabled
                <br />
                <input id="radioAppKeyboardEnabled" type="radio" name="keyboardType" /> Application screen keyboard
                enabled
                <br />
                <br />
                Automatic keyboard position
                <br />
                <select id="keyboardPosition">
                <option value="KeyboardPositionAuto">Auto</option>
                <option value="KeyboardPositionLeft">Left side</option>
                <option value="KeyboardPositionTopLeft">Top
                •                                                                                    •
                left corner</option>
                <option value="KeyboardPositionTop">Top side</option>
                <option value="KeyboardPositionTopRight">Top
                •                                                                                    •
                right corner</option>
                <option value="KeyboardPositionRight">Right side</option>
                <option value="KeyboardPositionBottomRight">Bottom
                •                                                                                    •
                right corner</option>
                <option value="KeyboardPositionBottom">Bottom side</option>
                <option value="KeyboardPositionBottomLeft">Bottom
                •                                                                                    •
                left corner</option>
                </select>
        </div>
</body>
</html>
```

global.magstripe Object

The global.magstripe object is a widget that provides an interface to magnetic card readers and barcode scanners. In the case of a card reader, the widget reacts to a scan of a card and returns the value of the data that is recorded on the magnetic stripe. For credit cards, the data returned is typically cardholder's name, card number, and expiration date. The widget returns the data in an unparsed form, so it is the responsibility of the developer to decrypt if necessary and parse the data. For a barcode reader, the widget registers a scanned event and returns the string that represents the barcode.

Refer to the latest *Moderro Interactive Experience Client 4600 Series User Guide* for a list of supported magnetic card readers and barcode scanners.

The global.magstripe object code is:

```
interface Magstripe {
        void open(in string deviceName); void
        close();

signals:
        void opened();
        void scanning();
        void scanned(out string data); void
        error(out string error);
}
```

Table 4-7 global.magstripe Object Variables

Variable	Description
open(in string deviceName)	Opens the device for reading data. If deviceName is not empty, use this device name, and browser.magstripe.scanner property otherwise.
close()	Closes the device
opened()	The device has been open successfully.
scanning()	The device has started data scanning.
scanned(out string data)	The device has finished scanning and read the scanned data from deviceName.
error(out string error)	An error has occurred.

To enable the magnetic card reader or barcode scanner, you will need the exact name of the card reader or barcode scanner by which the IEC recognizes the peripheral. Follow these steps to retrieve that name:

Step 1 Plug the magnetic card reader or barcode scanner into the USB port of the IEC.

Step 2 Reboot the IEC so that the IEC will recognize the new peripheral.

Step 3 Run the **lsinput**command at the shell prompt to get a list of connected input devices.

Note Alternatively, you can get the name from the Moderro Interactive Experience Manager (IEM). Go to the device and click the Status tab.

Step 4 Find the name of the magnetic card reader or barcode scanner. In the example below, the magnetic card reader is "Mag-Tek USB Swipe Reader".

```
Virtual core pointer id=2[master pointer   (3)]
      Virtual core XTEST pointer id=4[slave   pointer   (2)]
      Microsoft Microsoft® Digital Media Keyboardid=12[slave pointer (2)] Filtered Elo TouchSystems, Inc.
      Elo TouchSystems 2700 IntelliTouch(r) USB Touchid=14[slave   pointer   (2)]
      MCE IR Keyboard/Mouse (ite-cir) id=15[slave   pointer   (2)]
      Elo TouchSystems, Inc. Elo TouchSystems 2700 IntelliTouch(r) USB Touchmonitor Interfaceid=9[slave   pointer   (2)]
Virtual core keyboard id=3[master keyboard (2)]
      Virtual core XTEST keyboard id=5[slave keyboard (3)] Power Button
      id=6[slave   keyboard (3)]
      Video Bus id=7[slave keyboard (3)] Power
      Button id=8[slave   keyboard (3)]
      PWC snapshot button id=10[slave   keyboard (3)]
      Microsoft Microsoft® Digital Media Keyboardid=11[slave keyboard (3)] Mag-Tek USB Swipe
      Reader id=13[slave   keyboard (3)]
      ACPI Virtual Keyboard Device id=16[slave keyboard (3)] ITE8704 CIR
      transceiver id=17[slave   keyboard (3)]
```

Step 5 Replace the deviceName variable in the global.magstripe object with the name of the peripheral that the IEC recognizes.

Step 6 In the device's profile or a property applied to that device within the IEM, configure the application data property with "barcode.scanner" or "magstripe.scanner" for the key and the name of the peripheral that the IEC recognizes for the value.

The following is an example global.magstripe usage (file ../../test/js/magstripe.html).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
      <title>..:: global .magstripe test ::.. </title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<s
tyl
e>
bo
dy
{
      margin: 20px;
      background-color: #111111;
      color : #eeeeee;
      font-weight: bold;
      font-family: Arial;
      font-size: 18px;
}
</style>

<script type="text/javascript"> var

nameId;
var dataId;
var errorId ;
var statusId
;

function init ()
{
      nameId = document.getElementById("name"); dataId =
      document.getElementById("data"); errorId =
```

```
        document.getElementById("error"); statusId =
        document.getElementById("status"); global .magstripe.error
        .connect(onError); global
        .magstripe.scanning.connect(onScanning); global
        .magstripe.scanned.connect(onScanned); global
        .magstripe.opened.connect(onOpened);
}
function sopen()
{
        errorId .innerHTML = "";
        statusId .innerHTML = ""; try
        {
                global .magstripe.open(nameId.value);
                }
        catch(ex)
        {
                alert ("Exception: " + ex);
        }
}
function sclose ()
{
try
{
   global .magstripe.close () ;
}
catch(ex)
{
   alert ("Exception: " + ex);
}
}
function onError(e)
{
   statusId .innerHTML = "Error"; errorId
   .innerHTML = e;
}
function onScanning()
{
   dataId.innerHTML = "";
        statusId .innerHTML = "Scanning";
}
function onScanned(data)
{
        dataId.innerHTML = data; statusId
        .innerHTML = "Scanned";
}
function onOpened()
{
        statusId .innerHTML = "Opened";
}
</script>
</head>
        <body onload="init()">
        Device name: <input id="name" type="text" />
        <input type="button" value="Open" onclick="sopen()" />
        <input type="button" value="Close" onclick="sclose()" /> <br><br> Incoming data: <span
        id="data"></span><br>
        Status: <span id="status"></span><br> Error: <span
        id="error"></span>
</body>
</html>
```

global.mediaCache Object

This object allows you to control the local media cache.

```
interface MediaCache
{
        readonly attribute long long capacity; readonly
        attribute long long size; readonly attribute long
        long free; readonly attribute bool isEmpty;
        readonly attribute int fileCount READ fileCount; readonly
        attribute Array urls;

        int clear(in long long size) const; bool
        contains(in string url) const; int load(in
        string url) const;   bool remove(in string url)
        const; bool isLocked(in string url) const;
        long long fileSize(in string url) const; Date
        created(in string url) const;
        Date lastAccessed(in string url) const; QString
        mimeType(in string url) const;

slots:

        int

clear(); signals:

        void overloaded();
        void loadStarted(in string url); void
        loadFinished(in string url);
        void loadError(in string url, in string errorString); void removed(in
        string url);
        void cleared();
        void locked(in string url);
        void unlocked(in string url);

};
```

Table 4-8 global.mediaCache Variables

Variable	Description
capacity	Cache capacity in bytes
size	Cache size (occupied space) in bytes
free	Available space in the cache in bytes
isEmpty	Returns true if there are no files in the cache
fileCount	Number of files in the cache
urls	List of cached URLs in the order of last access date and time. Files accessed earlier go first.
clear(in long long size)	Removes the most rarely used files to get at least size bytes available in the cache.
contains(in string url)	Returns true if the cache has the result of HTTP GET request to URL url cached.
load(in string url)	Loads the specified URL into the cache.
remove(in string url)	Removes the file correspondent to URL url from the cache.

isLocked(in string url)	Returns true if the file correspondent to URL url is in use.
fileSize(in string url)	Returns size in bytes for the file correspondent to URL url.
created(in string url)	Returns creation date and time for the file correspondent to URL url.
lastAccessed(in string url)	Returns last access date and time for the file correspondent to URL url.
mimeType(in string url)	Returns MIME type for the file correspondent to URL url.
clear()	Clears the cache. All the files (including locked ones) will be removed from the cache.

All the signals purpose is clear.

Note You should use clear() method very carefully since it can put your application in an inconsistent state.

For example, it can cause playback errors for the video player.

The following HTML contains an example of global.mediaCache usage (file ../../test/js/mediacache.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>

    <title>..:: global.mediaCache test ::..</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<style>

body
{
    margin: 20px;
    background-color: #000000;
    color: #eeeeee;
    font-weight: bold;
    font-family: Arial;
    font-size: 20px;
}

</style>

<script type="text/javascript"> var

textUrl;

function flag(f)
{
    return f ? "Yes" : "No";
}

function updateCacheStatus()
{
    document.getElementById("isEmpty").innerHTML = flag(global.mediaCache.isEmpty);
    document.getElementById("capacity").innerHTML = global.mediaCache.capacity;
    document.getElementById("size").innerHTML = global.mediaCache.size; document.getElementById("free").innerHTML =
    global.mediaCache.free; document.getElementById("fileCount").innerHTML = global.mediaCache.fileCount;
    document.getElementById("urls").innerHTML = global.mediaCache.urls.join("<br />");
}
```

```
function init()
{
        textUrl         =         document.getElementById("textUrl");
        global.mediaCache.removed.connect(onRemoved);
        updateCacheStatus();
}

function check()
{
        var url = textUrl.value;
        var contains = global.mediaCache.contains(url); document.getElementById("contains").innerHTML = flag(contains);
        document.getElementById("mimeType").innerHTML = contains ? global.mediaCache.mimeType(url) : " ";
        document.getElementById("fileSize").innerHTML = contains ? global.mediaCache.fileSize(url) : " ";
        document.getElementById("created").innerHTML = contains ? global.mediaCache.created(url).toLocaleString() :
        " "; document.getElementById("lastAccessed").innerHTML = contains ?
        global.mediaCache.lastAccessed(url).toLocaleString() : " "; document.getElementById("isLocked").innerHTML =
        contains ? flag(global.mediaCache.isLocked(url)): " ";
}

function remove()
{
          var url = textUrl.value;
          global.mediaCache.remove(url);
          updateCacheStatus();
}
function onRemoved(url)
{
        document.getElementById("removed").innerHTML = url;
}

</script>

</head>

 <body onload="init()">

        <table cellpadding="2" cellspacing="0" border="1px" align="center" width="100%">
                <tr>
                <td width="50%">Is empty</td>
                <td id="isEmpty"></td>
                </tr>
                <tr>
                <td>Capacity in bytes</td>
                <td id="capacity"></td>
                </tr>
                <tr>
                <td>Size in bytes</td>
                <td id="size"></td>
                </tr>
                <tr>
                <td>Free space in bytes</td>
                <td id="free"></td>
                </tr>
                <tr>
                <td>Number of files</td>
                <td id="fileCount"></td>
                </tr>
                <tr>
                <td>
                URL to check
```

```
                <input id="textUrl" type="text" />
                </td>
                <td>
                <input type="button" value="Check" onclick="check()" />
                <input type="button" value="Remove" onclick="remove()" />
                </td>
                </tr>
                <tr>
                <td>Exists</td>
                <td id="contains"></td>
                </tr>
                <tr>
                <td>MIME type</td>
                <td id="mimeType"></td>
                </tr>
                <tr>
                <td>File size in bytes</td>
                <td id="fileSize"></td>
                </tr>
                <tr>
                <td>Creation time</td>
                <td id="created"></td>
                </tr>
                <tr>
                <td>Last access time</td>
                <td id="lastAccessed"></td>
                </tr>
                <tr>
                <td>Is locked</td>
                <td id="isLocked"></td>
                </tr>
                <tr>
                <td>URL removed</td>
                <td id="removed"></td>
                </tr>
                <tr>
                <td valign="top">URLs in the Cache</td>
                <td id="urls"></td>
                </tr>
        </table>

</body>
</html>
```

The following IEM properties are connected to media cache:

- browser.cache.media.enabled: Specifies whether media cache is enabled. You must set this property to 'true' to turn on the media cache.
- browser.cache.media.size: Specifies media cache size in megabytes (default is 2048). Must not be set to zero in order to turn on media content caching. It is not recommend to use a size less than 50 megabytes.
- browser.cache.media.path: Specifies the path where the cached content will be stored. You do not need to change this property unless you want to place the cache on an external storage connected via USB.
- browser.cache.media.mode: Specifies the caching mode. This is the most important property for media cache. Here is the list of values you can set for this property:
  - MediaCacheCheckIfExpired: Default value. Every time your application requests some

media resource, this value will tell the IEC to check if the content has expired. If this is the case, the content will be re-cached. If not, the cached content will be returned.

– MediaCacheNeverExpired: If content is cached, this value tells the IEC to never check if it's expired (i.e. it is in offline mode). In contrast to the 'CacheAlwaysCache' value for browser.cache.web.mode, media cache in this mode will try to load uncached content from the network.

global.network Object

Network information can be obtained using global.network global object. It implements the Network interface.

```
interface Network
 {
        readonly attribute string ifName; readonly
        attribute bool isWireless; readonly attribute
        string ip; readonly attribute string mask;
        readonly attribute string mac;
        readonly attribute string defaultGateway; readonly
        attribute string[] dnsServers;
 };
```

Table 4-9 global.network Variables

Variable	Description
ifName	Active network interface name (e.g. 'eth0' or 'wan0')
isWireless	If true, the wireless network interface is active.
ip	Active network interface IP address
mask	Active network interface subnet mask
mac	Active network interface MAC address
defaultGateway	Default gateway IP address
dnsServers	List of DNS servers IP addresses

The following HTML is an example of global.network usage:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose. dtd">
        <html>

        <head>

        <title>..:: global.network test ::..</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

        <style>

        body
        {
        margin: 20px;
        background-color: #000000;
      color: #eeeeee;
      font-weight: bold;
      font-family: Arial;
```

```
        font-size: 20px;
}


        </style>

        <script type="text/javascript"> var

        stockClip;
        var weatherClip;

        function init()
        {
        document.getElementById("ifName").innerHTML = global.network.ifName;
        document.getElementById("isWireless").innerHTML = global.network.isWireless ?
"Yes" : "No";
        document.getElementById("ip").innerHTML = global.network.ip; document.getElementById("mask").innerHTML =
        global.network.mask; document.getElementById("mac").innerHTML = global.network.mac;
        document.getElementById("defaultGateway").innerHTML =
global.network.defaultGateway;
        document.getElementById("dnsServers").innerHTML =
global.network.dnsServers.join(", ");
        }

        </script>
        </head>

        <body onload="init()">

        <table cellpadding="2" cellspacing="0" border="1px" align="center" width="50%">
        <tr>
        <td>Interface name</td>
        <td id="ifName"></td>
        </tr>
        <tr>
        <td>Is a wireless interface</td>
        <td id="isWireless"></td>
        </tr>
        <tr>
        <td>IP address</td>
        <td id="ip"></td>
        </tr>
        <tr>
        <td>Network mask</td>
        <td id="mask"></td>
        </tr>
        <tr>
        <td>MAC address</td>
        <td id="mac"></td>
        </tr>
        <tr>
        <td>Default gateway</td>
        <td id="defaultGateway"></td>
        </tr>
        <tr>
        <td>DNS servers</td>
        <td id="dnsServers"></td>
        </tr>
        </table>

        </body>

  </html>
```

global.networkCache Object

Similar to the global.mediaCache object, the global.networkCache object allows you to control the local network cache.

```
interface networkCache
{
        readonly attribute bool isEnabled; readonly
        attribute long long capacity; readonly attribute
        long long size; readonly attribute long long free;
        readonly attribute bool isEmpty;

        bool contains(in string url) const; bool
        remove(in string url) const;
        Date lastModified(in string url) const; slots:

          int clear();

};
```

Table 4-10 global.networkCache Variables

Variable	Description
isEnabled	Returns true if the network cache is enabled or returns false if otherwise
capacity	Cache capacity in bytes
size	Cache size (occupied space) in bytes
free	Available space in the cache in bytes
isEmpty	Returns true if there are no files in the cache
contains(in string url)	Returns true if the cache has the result of HTTP GET request to URL cached
remove(in string url)	Removes the file correspondent to URL from the cache
lastModified(in string url)	Returns last modification date and time for the file correspondent to URL
clear()	Clears the cache: All the files (including locked ones) will be removed from the cache.

The following HTML contains an example of global.networkCache usage (file../../test/js/networkcache.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
        <html>

        <head>

        <title>..:: global.networkCache test ::..</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

        <style>
```

```
body
{
margin: 20px;
background-color: #000000;
color: #eeeeee;
font-weight: bold;
font-family: Arial;
font-size: 20px;
}

</style>

<script type="text/javascript"> var

textUrl;

function flag(f)
{
return f ? "Yes" : "No";
}
function updateCacheStatus()
{
document.getElementById("isEmpty").innerHTML = flag(global.networkCache.isEmpty);
document.getElementById("capacity").innerHTML = global.networkCache.capacity;
document.getElementById("size").innerHTML = global.networkCache.size;
document.getElementById("free").innerHTML = global.networkCache.free;
}

function init()
{
textUrl = document.getElementById("textUrl");
updateCacheStatus();
}

function check()
{
var url = textUrl.value;
var contains = global.networkCache.contains(url); document.getElementById("contains").innerHTML = flag(contains);
document.getElementById("lastModified").innerHTML = contains ?
global.networkCache.lastAccessed(url).toLocaleString() : " ";
}

function remove()
{
var url = textUrl.value;
global.networkCache.remove(url);
updateCacheStatus();
}

</script>

</head>

<body onload="init()">

<table cellpadding="2" cellspacing="0" border="1px" align="center" width="100%">
<tr>
<td width="50%">Is empty</td>
<td id="isEmpty"></td>
</tr>
<tr>
<td>Capacity in bytes</td>
<td id="capacity"></td>
```

```
</tr>
<tr>
<td>Size in bytes</td>
<td id="size"></td>
</tr>
<tr>
<td>Free space in bytes</td>
<td id="free"></td>
</tr>
<tr>
<td>
URL to check
<input id="textUrl" type="text" />
</td>
<td>
<input type="button" value="Check" onclick="check()" />
<input type="button" value="Remove" onclick="remove()" />
</td>
</tr>
<tr>
function updateCacheStatus()
{
document.getElementById("isEmpty").innerHTML = flag(global.networkCache.isEmpty);
document.getElementById("capacity").innerHTML = global.networkCache.capacity;
document.getElementById("size").innerHTML = global.networkCache.size;
document.getElementById("free").innerHTML = global.networkCache.free;
}

function init()
{
textUrl = document.getElementById("textUrl");
updateCacheStatus();
}

function check()
{
var url = textUrl.value;
var contains = global.networkCache.contains(url); document.getElementById("contains").innerHTML = flag(contains);
document.getElementById("lastModified").innerHTML = contains ?
global.networkCache.lastAccessed(url).toLocaleString() : " ";
}

function remove()
{
var url = textUrl.value;
global.networkCache.remove(url);
updateCacheStatus();
}

</script>

</head>

<body onload="init()">

<table cellpadding="2" cellspacing="0" border="1px" align="center" width="100%">
<tr>
<td width="50%">Is empty</td>
<td id="isEmpty"></td>
</tr>
<tr>
<td>Capacity in bytes</td>
<td id="capacity"></td>
```

```
</tr>
<tr>
<td>Size in bytes</td>
<td id="size"></td>
</tr>
<tr>
<td>Free space in bytes</td>
<td id="free"></td>
</tr>
<tr>
<td>
URL to check
<input id="textUrl" type="text" />
</td>
<td>
<input type="button" value="Check" onclick="check()" />
<input type="button" value="Remove" onclick="remove()" />
</td>
</tr>
<tr>
<td>Exists</td>
<td id="contains"></td>
</tr>
<tr>
<td>Last modified</td>
<td id="lastModified"></td>
</tr>
</table>

</body>

</html>
```

The following properties are related to network cache:

Table 4-11 global.networkCache Properties

Variable	Description
browser.cache.web.enabled	Specifies whether web cache is enabled. This property must be set to true to turn on web content caching.
browser.cache.web.size	Specifies web cache size in megabytes (default is 1024). Must not be zero to turn on web content caching. It is not recommend that you use a size less than 50 megabytes.
browser.cache.web.path	Specifies the path, where the cached content will be stored. You do not need to change this property, unless you want to place the cache on an external storage connected via USB.
browser.cache.media.clear	Once you set this property to true, the media cache is cleared. If you read value of this property, you always get false.

browser.cache.web.mode	This is the caching mode. This is the most important property for web cache. Here is the list of values you can set for this property: • CacheAlwaysNetwork: Always load from network and do not check if the cache has a valid entry (similar to the Reload feature in browsers). • CachePreferNetwork: Default value. Load from the network if the cached entry is older than the network entry. • CachePreferCache: Load from cache if available, otherwise load from network. Note that this can return possibly stale (but not expired) items from cache. • CacheAlwaysCache: Only load from cache, indicating error if the item was not cached (i.e., off-line mode).

If you need to cache some web content to be able to use it later in off-line mode, you have to load that content with browser.cache.web.mode set to any value but "CacheAlwaysCache". After that from your application you have to set this property to "CacheAlwaysCache" or “CachePreferCache”. Note that in that mode if you did not cache some content your application requires, that content will not be loaded from the network when your application requests it. Instead you will see the "Service temporarily unavailable" error message.

global.printer Object

The global.printer object implements a printer interface which allows control of the printer connected to the IEC4600 Series either locally or via the network. This object allows end users to print PDF files, images, plain text documents and HTML documents. Plain text and HTML documents must be UTF-8 encoded in order to be printed correctly.

Refer to the latest *Moderro Interactive Experience Client 4600 Series User Guide* for a list of supported printers and instructions on how to set up and test your printer.

Note While printing HTML documents, the end user will not be able to print external resources referred by that document such as images, flash clips or plugins.

The following is the global.printer object code:

```
interface Printer
        {
            attribute bool collateCopies; readonly
            attribute int colorCount; attribute
            ColorMode colorMode; attribute int
            copyCount;
            attribute bool doubleSidedPrinting; attribute
            DuplexMode duplex; attribute bool
            fontEmbeddingEnabled; readonly attribute int
            fromPage; readonly attribute int toPage;
            attribute bool fullPage;
            readonly attribute int widthMM; readonly
            attribute int heightMM; readonly attribute bool
            isValid; readonly attribute string name; attribute
            Orientation orientation; attribute PageOrder
            pageOrder; attribute PaperSize paperSize;
            attribute PaperSource paperSource; readonly
            attribute PrinterState state; attribute int
            resolution;
```

```
        readonly attribute list <int> supportedResolutions; readonly attribute list
        <PaperSize> supportedPaperSizes; readonly attribute bool
        supportsMultipleCopies;
        readonly attribute list <string> availablePrinters; readonly attribute string
        defaultPrinter;
        readonly attribute map status;

        slot bool abort(); slot
        bool newPage();
        slot bool clearJobQueue();

        list <real> getPageMargins(in Unit unit) const;
        void setPageMargins(in real left, in real top, in real right, in real bottom, in Unit unit);
        bool setCurrentPrinter(in string printerName); list <real>
        paperExactSize(Unit unit) const;
        void setPaperExactSize(in real width, in real height, Unit unit) const; int print(in string url);
        int printCurrentPage();
        int printCurrentPageRect(in int left , in int top, in int width, in int height); int printElementBySelector(in string
        cssSelector );

        void setFromTo(in int from, in int to);


        void clearStatusHistory();

    signals:
        void errorStatus(in Date date, in int code, in string errorString);
    }
```

Table 4-12 global.printer Variables

Variable	Description
collateCopies	Contains true if collation is turned on when multiple copies is selected. Contains false if it is turned off when multiple copies is selected. When collating is turned off the printing of each individual page will be repeated the numCopies amount before the next page is started. With collating turned on all pages are printed before the next copy of those pages is started.
colorCount	Contains the number of different colors available for the printer
colorMode	Contains the current color mode
copyCount	Contains the number of copies that will be printed: The default value is 1.
doubleSidedPrinting	Contains true if double side printing is enabled
duplex	Contains the current duplex mode
fontEmbeddingEnabled	Contains true if font embedding is enabled
fromPage	Contains the number of the first page in a range of pages to be printed. Pages in a document are numbered according to the convention that the first page is page 1. By default, this attribute contains a special value of 0, meaning that the "from page" setting is unset. This is read-only attribute. Use the setFromTo() method to set page range.

toPage	Contains the number of the last page in a range of pages to be printed. Pages in a document are numbered according to the convention that the first page is page 1. By default, this attribute contains a special value of 0, meaning that the "to page" setting is unset. This is a read-only attribute. Use the setFromTo() method to set page range.
fullPage	Contains true if the origin of the printer's coordinate system is at the corner of the page and false if it is at the edge of the printable area
widthMM	Contains the width of printing area in millimeters
heightMM	Contains the height of printing area in millimeters
isValid	Contains true if the printer currently selected is a valid printer in the system
name	Contains the printer name
orientation	Contains the orientation setting
pageOrder	Contains the current page order
paperSize	Contains the printer paper size
paperSource	Contains the printer's paper source
printerState	Contains the current state of the printer: This may not always be accurate (for example if the printer doesn't have the capability of reporting its state to the operating system).
resolution	Contains the current assumed resolution of the printer
supportedResolutions	Contains a list of the resolutions (a list of dots-per-inch integers) that the printer says it supports
supportedPaperSizes	Contains a list of paper sizes that the printer says it supports.
supportsMultipleCopies	Returns true if the printer supports printing multiple copies of the same document in one job; otherwise false is returned. On most systems this function will return true.
availablePrinters	Contains a list of names of supported printers connected to the device
defaultPrinter	Contains a name of default printer.
status	Printer status as reported by the driver. Works only with HP printers. For example, it returns ‘1014’ for a paper jam or ‘1009’ when the printer is out of paper.
abort()	Aborts the current print run. Returns true if the print run was successfully aborted and printerState will return Printer:Aborted; otherwise returns false. It is not always possible to abort a print job. For example, all the data has gone to the printer but the printer cannot or will not cancel the job when asked to do.
newPage()	Tells the printer to eject the current page and to continue printing on a new page. Returns true if this was successful; otherwise returns false.
clearJobQueue()	Tells the printer to cancel all print jobs. Returns true on success; otherwise returns false.
getPageMargins()	Returns the page margins for this printer for the left, top, right, and bottom margins. The unit of the returned margins are specified with the unit parameter.
setPageMargins()	Sets the left, top, right and bottom page margins for this printer. The unit of the margins are specified with the unit parameter.

setCurrentPrinter()	Sets the printer identified by its name as a current printer. Returns true on success, false on failure. List of all printer's names can be retrieved using availablePrinters attribute. Initially defaultPrinter is considered current.
paperExactSize()	Returns the paper size as an array of two real numbers for page width and height in specified length unit.
setPaperExactSize()	Sets the paper width and height in specified length unit.
print()	Prints document given by its URL. The URL can be local file system path or an HTTP URL. Returns PrintJob object.
printCurrentPage()	Prints web page currently opened in a browser. Returns PrintJob object.
printCurrentPageRect()	Prints rectangle of the current web page defined by left, top, width and height arguments. Returns PrintJob object.
printElementBySelector()	Prints first element matching given CSS selector. Returns PrintJob object.
setFromTo()	Sets the range of pages to be printed. Pages in a document are numbered according to the convention that the first page is page 1. All pages will be printed if the both arguments are 0.
clearStatusHistory()	Clears the queue with the printer status history.
errorStatus()	Fires when an error status event has arrived from the printer.

All global.printer methods related to performing actual printing return objects implementing the PrintJob interface.

```
interface PrintJob {
        readonly attribute JobState state; readonly
        attribute string errorString; readonly attribute
        string printerName; readonly attribute bool
        isFinished;

        void
        cancel();
        void
        remove();

        signal void finished();
        signal void error();
        signal void stateChanged();
}
```

Table 4-13 PrintJob Variables

Variable	Description

state	Contains current state of the print job and has one of the following values: 'Downloading' — document is being downloaded from remote server 'Held' — job is held for printing 'Pending' — job is waiting to be printed 'Processing' — job is currently printing 'Completed' — job has completed successfully 'Stopped' — job has been stopped 'Aborted' — job has aborted due to an error
errorString	Contains string describing the error that has occurred
printerName	Contains name of a printer performing the job
isFinished	Contains 'true' if printer finished processing the job regardless if it was successful or not or 'false' if printer has not finished processing the job
cancel()	Tells the printer to cancel processing the print job. Returns 'true' on success or 'false' on failure. Note that it is not always possible to stop printing immediately if this process already started.
remove()	Tells the browser to remove job object. Information about all processed and finished jobs are kept in memory. If application uses printing extensively, it may be necessary to free resources associated with finished jobs manually using this method. Job object should never be used after the call of this method.
finished()	Returns when the job is finished processing regardless if it was successful or not
error()	Returns when error related to the job occurs
stateChanged()	Returns every time job state changes

The following HTML contains an example of global.printer usage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>

<title>..:: global.printer test ::..</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<s

ty

le

>

b

o

dy
{
        margin: 20px;
        background-color: #000000;
        color: #eeeeee;
```

```
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
}

</style>
<script type="text/javascript">
function init ()
{
        var printerSelect = document.getElementById("printer"); var printerList =
        global. printer . availablePrinters ; var defaultPrinter = global. printer .
        defaultPrinter ; for (var i = 0; i < printerList .length; ++i) {
                var printer = printerList [ i ];
                printerSelect [ i ] = new Option(printer, printer); if ( printer ==
                defaultPrinter)
                printerSelect .selectedIndex = i;
        }
        printerChanged();
        global . printer . finished .connect(onFinished); global . printer
        .pagePrinted.connect(onPagePrinted);
}

function printerChanged()
{
        var printerSelect = document.getElementById("printer");

        var printerName = printerSelect[printerSelect .selectedIndex ]. value; global . printer .setCurrentPrinter(printerName);
        document.getElementById("resolution").innerHTML =global.printer.resolution;
        document.getElementById("state").innerHTML =global.printer.state;

}

function onFinished(ok)
{
        document.getElementById("jobState").innerHTML = ok
                ? "Succeeded"
                : ("Failed: " + global. printer . errorString ) ;
}

function onPagePrinted(number)
{
        document.getElementById("pageNumber").innerHTML = number;
}

function print ()
{
        global . printer . print("http://embedded-computing.com/pdfs/QNX.Jan05.pdf");
}
</script>
</head>
 <body onload="init()">
         <form onsubmit="print()">
                <table cellpadding="2" cellspacing="0" border="1" align="center" width="50%">
                <tr>
                <td>Printer</td>
                <td><select id="printer" onchange="printerChanged()"></select></td>
                </tr>
                <tr>
                <td>Resolution</td>
                <td id="resolution"></td>
                </tr>
                <tr>
                <td>State</td>
```

```
            <td id="state"></td>
            </tr>
            <tr>
            <td>Page printed</td>
            <td id="pageNumber"></td>
            </tr>
            <tr>
            <td>Job status</td>
            <td id="jobState"></td>
            </tr>
            <tr>
            <td colspan="2" align="center">
            <input type="submit" value="Print" />
            </td>
            </tr>
            </table>
        </form>
</body>
</html>
```

See the *Moderro Interactive Experience Client 4600 Series User Guide* for more examples, tips, and testing information related to the global.printer object.

global.registry Object

This object implements the Registry interface. It allows you to get and set properties values. It also allows you to get and set persistent values in the device profile.

Note Wrong use of this API can be very dangerous. For example, wrong values for network-related properties can make your device inaccessible. In the worst case, you will have to re-burn the firmware of your device with an emergency USB stick.

Note When you use this object first time it will take several seconds for it to be initialized, since it has to get the information on all properties in the system.

```
interface Registry
{
    readonly attribute string[] properties; readonly
    attribute string module; readonly attribute bool
    isStandalone; readonly attribute string
    serviceHost;
    readonly attribute unsigned int servicePort; readonly
    attribute bool isDmmEnabled; readonly attribute string
    dmmUrl;
    readonly attribute int lastUpdated;
    readonly attribute int expiresIn;

    bool exists(in string propertyName) const; string
    value(in string propertyName) const;
    int setValue(in string propertyName,in string propertyValue) const;

    string title(in string propertyName) const; string description(in
    string propertyName) const;
```

```
        bool isPersistent(in string propertyName) const; bool
        isGlobal(in string propertyName) const; bool isInternal(in
        string propertyName) const; bool isRunTime(in string
        propertyName) const; bool isIpc(in string propertyName)
        const;
        bool isReadable(in string propertyName) const; bool
        isWritable(in string propertyName) const;

        string persistentValue(in string propertyName) const;
        int setPersistentValue(in string propertyName, in string propertyValue) const; bool subscribe(in string
        propertyName);

signals:
        void changed(in string propertyName);
};
```

Table 4-14 global.registry Variables

Variable	Description
properties	Returns names for all available properties in the system
module	Returns module name for the browser process
isStandalone	Returns true if the device is in stand-alone mode (is not registered in the IEM)
serviceHost	IEM host name or IP address
servicePort	IEM port
isDmmEnabled	Returns true if Cisco DMM support is turned on
dmmUrl	Returns Cisco DMM URL
lastUpdated	Returns time stamp in seconds since the beginning of Epoch (1-Jan-1970,12:00 AM), when the registry was updated from the management server.
expiresIn	Returns time in seconds after that amount of time the registry has to be considered as expired and has to be refreshed from the management server side. If contains a negative number or zero, the registry will not expire. Note that this value is not counted down as time goes. It updates every time the registry updates from the management server.
exists()	Returns true if the property with name propertyName exists
value()	Returns string serialization for the value of property propertyName. Note that if the property type is not integral (something like struct or list) this method returns XML.
propertyName.	Note that if the property type is not integral (e.g. struct or list) this method returns XML.
setValue()	Sets value for property propertyName. If the property type is not integral, you must provide an XML serialization as a property value. Returns zero on success, or -1 on error.
title()	Returns property title
description()	Returns property description
isPersistent()	Returns true if the property is persistent. That means its value will

	be obtained from the persistent storage at the module startup and it'll be saved to the persistent storage every time the property value changes.
isGlobal():	Returns true if the property is global. If property is global that means it has only one persistent value for all modules and it appears in the spec only once. Its full qualified name does not contain module section.
isInternal():	Returns true if the property is internal. This property cannot be accessed from a module this property does not belong to.
isRunTime():	Returns true if the property is run time. Changing this property's value implies immediate action.
isIpc():	Returns true if the property is used for IPC. It's an internal property used for IPC purposes only. So, it has to be ignored by management systems.
isReadable():	Returns true if the property value can be read
isWritable():	Returns true if the property value can be changed
persistentValue():	Returns string serialization for the value of property propertyName from the persistent storage
setPersistentValue():	Sets value for property propertyName in the persistent storage
subscribe()	Subscribes global.registry object to monitor propertyName property changes. When you subscribe global.registry to monitor some property, signal changed() fires every time property value changes. Note that subscriptions persist even if you unload the page where you called the subscribe() method.
changed()	Fires when value of property propertyName changes. It fires only if you previously called the subscribe() method with that property name.

The following HTML contains an example of global.registry usage:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>

<title>..:: global.registry test ::..</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<s

ty

le

>

b

o

dy
{
        margin: 20px;
        background-color: #000000;
        color: #eeeeee;
```

```
        font-weight: bold;
        font-family: Arial;
        font-size: 20px;
}

</style>

<script type="text/javascript"> var

textPropertyName;
var textPropertyValue;

function init()
{
        textPropertyName = document.getElementById("textPropertyName"); textPropertyValue =
        document.getElementById("textPropertyValue"); document.getElementById("module").innerHTML =
        global.registry.module; document.getElementById("isStandalone").innerHTML =
        flag(global.registry.isStandalone); document.getElementById("serviceHost").innerHTML =
        global.registry.serviceHost; document.getElementById("servicePort").innerHTML =
        global.registry.servicePort;
        document.getElementById("properties").innerHTML = global.registry.properties.join(", ");
        global . registry .changed.connect(onPropertyChanged);
}

function flag(f)
{
        return f ? "Yes" : "No";
}
function get()
{
        var name = textPropertyName.value; textPropertyValue.value =
        global.registry.value(name);
        document.getElementById("exists").innerHTML = flag(global.registry.exists(name));
        document.getElementById("title").innerHTML = global.registry.title(name);
        document.getElementById("description").innerHTML =global.registry.description(name);
        document.getElementById("isPersistent").innerHTML = flag(global.registry.isPersistent(name));
        document.getElementById("isGlobal").innerHTML =flag(global.registry.isGlobal(name));
        document.getElementById("isInternal").innerHTML = flag(global.registry.isInternal(name));
        document.getElementById("isRunTime").innerHTML = flag(global.registry.isRunTime(name));
        document.getElementById("isIpc").innerHTML = flag(global.registry.isIpc(name));
        document.getElementById("isReadable").innerHTML = flag(global.registry.isReadable(name));
        document.getElementById("isWritable').innerHTML = flag(global.registry.isWritable(name));
{

function set()
{
        global.registry.setValue(textPropertyName.value, textPropertyValue.value);
}

function subscribe()
{
        global . registry .subscribe(textPropertyName.value);
}

</script>

</head>

 <body onload="init()">

        <table cellpadding="2" cellspacing="0" border="1px" align="center" width="100%">
```

```
<tr>
<td>Module</td>
<td id="module"></td>
</tr>
<tr>
<td>Is standalone mode</td>
<td id="isStandalone"></td>
</tr>
<tr>
<td>Management Service host</td>
<td id="serviceHost"></td>
</tr>
<tr>
<td>Management Service port</td>
<td id="servicePort"></td>
</tr>
<tr>
<td>Property changed recently</td>
<td id="isDmmEnabled"></td>
</tr>
<tr>
<td>DMM URL</td>
<td id=”changed”></td>
</tr>
<tr>
<td>

Property name
<input id="textPropertyName" type="text" />
</td>
<
t
d
>
V
a
l
u
e
<input id="textPropertyValue" type="text" />
<input type="button" value="Get" onclick="get()" />
<input type="button" value="Set" onclick="set()" />
</td>
</tr>
<tr>
<td>Exists</td>
<td id="exists"></td>
</tr>
<tr>
<td>Title</td>
<td id="title"></td>
</tr>
<tr>
<td>Description</td>
<td id="description"></td>
</tr>
<tr>
<td>Is persistent</td>
<td id="isPersistent"></td>
</tr>
<tr>
<td>Is global</td>
<td id="isGlobal"></td>
```

```
                </tr>
                <tr>
                <td>Is internal</td>
                <td id="isInternal"></td>
                </tr>
                <tr>
                <td>Is run time</td>
                <td id="isRunTime"></td>
                </tr>
                <tr>
                <td>Is IPC</td>
                <td id="isIpc"></td>
                </tr>
                <tr>
                <td>Is readable</td>
                <td id="isReadable"></td>
                </tr>
                <tr>
                <td>Is writable</td>
                <td id="isWritable"></td>
                </tr>
                <tr>
                <td valign="top">All available properties</td>
                <td id="properties"></td>
                </tr>
        </table>

</body>

</html>
```

global.resources Object

```
interface Resources
{
        readonly attribute int cpuLoad; // CPU load in percents.
        readonly attribute List<double> cpuFrequency; // Instant frequency in MHz per core.

        readonly attribute int memoryAllocated; // Available memory in percents. readonly attribute long
        long memoryTotal; // RAM size in bytes.


        readonly attribute int storageAllocated; // Occupied room in data partition in percents.
        readonly attribute unsigned long long storageFree; // Available room in data partition in bytes.
        readonly attribute unsigned long long storageTotal; // Capacity of the data partition in bytes.
        readonly attribute unsigned long long storageCapacity; // Capacity of the entire SSD in bytes.
};
```

global.scanner Object

The global.scanner object implements an interface for optical scanners allowing an application displayed on a kiosk to scan and manipulate a document. Refer to the latest Moderro Interactive Experience Client 4600 Series User Guide for a list of supported optical scanners and instructions on how to set up and test your scanner.

The scanner library used is from SANE and the list of compatible devices can be found here: http://www.sane-project.org/sane-supported-devices.html

The global.scanner object code is:

```
interface Scanner
```

```
{
        attribute uint dpiX;
        attribute uint dpiY;
        attribute bool color;
        attribute string source;
        readonly attribute List<String> devices; readonly
       attribute List<String> sources; readonly attribute
       string lastError; readonly attribute string
       base64Data; readonly attribute bool busy;

        void setCurrentScanner(in string deviceName); signals:
        void finished();
        void error(out string error);

slots:
        start();
        stop();
        shutdown();
};
```

Table 4-15 global.scanner Object Variables

Variable	Description
dpiX	DPI X of the selected scanner
dpiY	DPI Y of the selected scanner
color	Selected scanner in color mode
source	Document source
devices	List of available scanners
sources	List of available document sources
lastError	Last error occurred
base64Data	Scanned image as base64 JPEG data
busy	Checks if the scanner is busy
setCurrentScanner(in string deviceName)	Sets the current scanner to use. You need to call this method before scanning.
finished()	Indicates that the scanner has finished scanning
error(out string error)	Indicates an error has occurred
start()	Starts scanning from the selected scanner and document source
stop()	Stops scanning
shutdown()	Shuts down scanning subsystem and resets all internal caches

Note Scanner hot plugging (or unplugging) is not supported by the application.

Tip If you plug in a new scanner, you will need to restart the application to use it.

The following HTML contains an example of global.scanner usage (file../../test/js/scanner.html):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>..:: global .scanner test ::.. </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<style>

body
{
        margin: 20px;
        background-color: #111111;
        color: #eeeeee;
        font-weight: bold;
        font-family: Arial;


       font-size: 18px;
}


</style>
<script type="text/javascript"> var
scannersId;
var errorId
; var
colorId ;
var dpixId;
var dpiyId;
var
sourceId;
var startId , stopId; var
imageId;
var error;

function init ()
{
        scannersId = document.getElementById("scanners"); errorId =
        document.getElementById("error"); dpixId =
        document.getElementById("dpix");
        dpiyId = document.getElementById("dpiy"); colorId =
        document.getElementById("color"); sourceId =
        document.getElementById("source"); startId =
        document.getElementById("start"); stopId =
        document.getElementById("stop"); imageId =
        document.getElementById("image");

        global .scanner.error .connect(onError)
        global .scanner. finished .connect(onFinished)

        error = false ;
}

function getDevices()
{
try
}
      var data = global.scanner.devices ;
      scannersId.options.length = 0;

      if (data != null && data.length)
```

```
    {
    scannersId.options.add(new Option("-", 0)); for (var i = 0;i <
    data.length;i++)
    {
    scannersId.options.add(new Option(data[i], i+1));
    }
    }
}
      catch(ex)
      {
            alert ("Exception: " + ex);
      }
}

function apply()
{
      if (global .scanner.setCurrentScanner(scannersId.options[scannersId.selectedIndex ]. text))
      {
            errorId .innerHTML = "Set";
            getDpiX();
            getDpiY();
            getColor();
            getSource();
            startId .disabled = false ;
            stopId.disabled = false ;
      }
      else
      {
            startId .disabled = true;
            stopId.disabled = true;
      }
}

function check()
Try}
}
      var available = global.scanner. isAvailable
      (scannersId.options[scannersId.selectedIndex ]. text) ; alert ("Available : " + (available
      ? "yes" : "no"));
}
      catch(ex)
      {
            alert ("Exception: " + ex);
      }
}

function start ()
{
try
}
      setSource() ;
      setDpiX();
      setDpiY();
      setColor() ;
      imageId.src = " file :///";   errorId .innerHTML =
      "Scanning..."; global .scanner. start () ;
}
      catch(ex)
      {
            alert ("Exception: " + ex);
      }
```

```
}

function stop()
{
        global .scanner.stop() ;
}

function getDpiX()
{
        dpixId.value = global.scanner.dpiX;
}

function setDpiX()
{
        global .scanner.dpiX = parseInt(dpix.value); getDpiX();
}

function getDpiY()
{
        dpiyId.value = global.scanner.dpiY;
}
function setDpiY()
{
        global .scanner.dpiY = parseInt(dpiy.value); getDpiY();
}

function getColor()
{
        colorId .checked = global.scanner.color ;
}

function setColor()
{
        global .scanner.color = colorId.checked; getColor();
}

function getSource()
{
Try
}
        var data = global.scanner.sources;
        sourceId.options.length = 0;
        sourceId.options.add(new Option("-", 0)); for (var i = 0;i <
        data.length;i++)
{
        sourceId.options.add(new Option(data[i], i+1));
}
var src = global.scanner.source; if (! src)
src = "-";
for (var i = 0;i < sourceId.length; i++)
{
if (sourceId.options[ i ]. text === src)
{
sourceId.options[ i ]. selected = true; break;
}
}
        catch(ex)
        {
                alert ("Exception: " + ex);
        }
```

```
}

function setSource()
{
	global .scanner.source = sourceId.options[sourceId.selectedIndex ]. text ; getSource();
}

function onError(err)
{
	errorId .innerHTML = err;
}

function onFinished()
{
errorId .innerHTML = "Done";
imageId.src = "" + global.scanner.base64Data;
}

</script>
</head>
<body onload="init()">
	Available scanners: <select size="1" id="scanners"></select>
	<input type="button" value="Detect scanners" onclick="getDevices()" />
	<input type="button" value="Check" onclick="check()" />
	<input type="button" value="Apply" onclick="apply()" /><br><br> DPI-X: <input
	id="dpix" type="text" />
	<input type="button" value="Get" onclick="getDpiX()" /><br><br> DPI-Y: <input
	id="dpiy" type="text" />
	<input type="button" value="Get" onclick="getDpiY()" /><br><br> Color: <input
	id="color" type="checkbox" />
	<input type="button" value="Get" onclick="getColor()" /><br><br> Source:
	<select size="1" id="source"></select><br><br>
	<input type="button" value="Start scan" id="start" disabled="true" onclick="start()"
	/>
	<input type="button" value="Stop scan" id="stop" disabled="true" onclick="stop()"
	/><br><br>
	Log: <span id="error"></span><br><br>
	<div style="height:1024px;width:1024px;overflow:scroll;">
	<img id="image"></img>
	</div>
</body>
</html>
```

See the *Moderro Interactive Experience Client 4600 Series User Guide* for more examples, tips, and testing information related to the global.scanner object.

global.serialPorts Object

global.serialPorts object is a factory for SerialPort objects. You can create multiple SerialPort objects in your application.

The following is the serialPort Object Interface:

```
readonly attribute stringlist availableDevices; readonly attribute intlist
standardBaudRates;

bool contains(in string deviceName) const; SerialPort
port(in string deviceName); bool deletePort(in Object
port);
bool deletePort(in string deviceName); void deleteAll
```

```
        ();

        attribute int baudRate; attribute int
        dataBits; attribute int parity;
        attribute int stopBits; attribute int
        flowControl; attribute int
        dataErrorPolicy;
        attribute bool dataTerminalReady;
        attribute bool requestToSend; readonly
        attribute int error;
        attribute bool settingsRestoredOnClose;

        ByteArray readAll();
        long long write(in ByteArray data);

signals:
        void aboutToClose();

        void bytesWritten(in long long bytes); void
        readChannelFinished();
        void readyRead();
        void baudRateChanged(in int baudRate, in int direction); void
        dataBitsChanged(in int dataBits);
        void parityChanged(in int parity); void
        stopBitsChanged(in int stopBits); void
        flowControlChanged(in int flow);
        void dataErrorPolicyChanged(in int policy); void
        dataTerminalReadyChanged(in bool set); void
        requestToSendChanged(in bool set); void error(in int
        serialPortError);
        void settingsRestoredOnCloseChanged(in bool restore);
```

The following is the serialPort usage in HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
 <html>

 <head>

 <title>..:: global.serialPorts test ::..</title>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

 <s

 tyl

 e>

 bo

 dy
 {
      margin: 20px;
      background-color: #000000;
      color: #eeeeee;
      font-weight: bold;
      font-family: Arial;
      font-size: 20px;
}

 </style>

 <script type="text/javascript"> var
```

```
deviceName;
var device;
var
dataRead;
var dataToWrite;


function read()
{
	var data = device.readAll(); var
	str = "";
	for (var i = 0; i < data.length; ++i)
		str += data[i] < 32 ? '.' : String.fromCharCode(data[i]); dataRead.value += str;
	device.write(data);
}

function send()
{
	device.write(dataToWrite.value);
	dataToWrite.value = "";
}

function apply()
{
	if (!deviceName.value.length)
		return;
	global . serialPorts . deleteAll () ;
	device = global. serialPorts .port(deviceName.value);
	device.baudRate = parseInt(document.getElementById("baudRate").value);
	document.getElementById("baudRate").value = device.baudRate; device.dataBits =
	parseInt(document.getElementById("dataBits").value);
	document.getElementById("dataBits").value = device.dataBits;
	device.flowControl = parseInt(document.getElementById("flowControl").value);
	document.getElementById("flowControl").value = device.flowControl;
	device. parity = parseInt(document.getElementById("parity").value);
	document.getElementById("parity").value = device.parity; device.stopBits =
	parseInt(document.getElementById("stopBits").value);
	document.getElementById("stopBits").value = device.stopBits; device.readyRead.connect(read);
}

function init ()
{
	global . serialPorts . deleteAll () ;
	deviceName = document.getElementById("deviceName"); var devices =
	global. serialPorts . availableDevices ; for (var i = 0; i < devices.length;
	++i)
	{
		var opt = document.createElement("option"); opt.value
		= devices[i ];
		opt.innerHTML = opt.value;
		deviceName.appendChild(opt);
	}
	var baudRates = global.serialPorts.standardBaudRates; for (var i = 0; i
	< baudRates.length; ++i)
	{
		var opt = document.createElement("option"); opt.value
		= baudRates[i];
		if (opt.value == 115200) opt.
		selected = true;
		opt.innerHTML = opt.value;
		baudRate.appendChild(opt);
```

```
        }
            dataRead = document.getElementById("dataRead"); dataToWrite =
            document.getElementById("dataToWrite");
        }

</script>
</head>
 <body onload="init()">
      <table cellpadding="2" cellspacing="0" border="1" align="center" width="50%">
            <tr>
            <td colspan="2" align="center">Serial port settings</td>
            </tr>
            <tr>
            <td>Device</td>
            <td>
            <select id="deviceName">
            <option value="">Not selected</option>
            </select>
            </td>
            </tr>
            <tr>
            <td>Baud rate</td>
            <td>
            <select id="baudRate" />
            </td>
            </tr>
            <tr>
            <td>Data bits</td>
            <td>
            <select id="dataBits">
            <option value="5">5</option>
            <option value="6">6</option>
            <option value="7">7</option>
            <option value="8" selected>8</option>
            </select>
            </td>
            </tr>
            <tr>
            <td>Flow control</td>
            <td>
            <select id="flowControl">
            <option value="0" selected>No</option>
            <option value="1">Hardware</option>
            <option value="2">Software</option>
            </select>
            </td>
            </tr>
            <tr>
            <td>Parity</td>
            <td>
            <select id="parity">
            <option value="0" selected>No</option>
            <option value="2">Even</option>
            <option value="3">Odd</option>
            <option value="4">Space</option>
            <option value="5">Mark</option>
            </select>
            </td>
            </tr>
            <tr>
            <td>Stop bits</td>
```

```
                <td>
                <select id="stopBits">
                <option value="1" selected>1</option>
                <option value="3">1.5</option>
                <option value="2">2</option>
                </select>
                </td>
                </tr>
                <tr>
                <td colspan="2" align="right">
                <input type="button" value="Apply" onclick="apply()" />
                </td>
                </table>
                <br />
                <table cellpadding="2" cellspacing="0" border="1" align="center" width="50%">
                <tr>
                <td>Data read</td>
                <td>
                <textarea id="dataRead" readonly ></textarea>
                </td>
                </tr>
                <tr>
                <td>Data to write</td>
                <td>
                <textarea id="dataToWrite" ></textarea>
                <br />
                <input type="button" value="Send" onclick="send()" />
                </td>
                </tr>
    </table>
</body>
</html>
```

global.system Object

This object implements system interface.

```
interface System
{
        readonly attribute int screen; readonly
        attribute int masterScreen;
        readonly attribute list <string> usbDevices; readonly attribute
        list<string, string> webCameras; attribute int idleTimeout;

        string envVariable(in string name) const;
        int setEnvVariable(in string name, in string value,in bool overwrite = true); int unsetEnvVariable(in string
        name);
        Screenshot screenshot(in int width, in int height);

        void debug(in string message) const; void
        info(in string message) const; void warning(in
        string message) const; void error(in string
        message) const;
        void moveMouse(in int x, in int y) const; void
        mouseClick(in int button = 0) const;

signals:
        void mouseDoubleClicked(in int x, in int y,in int button); void mouseMoved(in
        int x, in int y,in int button);
        void mousePressed(in int x, in int y,in int button); void
        mouseReleased(in int x, in int y,in int button);
```

```
    void mouseWheelRotated(in int x, in int y, in int button); void idle();
    void usbConnected(in string idAndName); void
    usbDisconnected(in string idAndName);
};
```

Table 4-16 global.system Variables

Variable	Description
screen	Returns screen number the browser started on. Using this value web application can show different content on different monitors.
masterScreen	Returns the number of screen showed on the master display.
usbDevices	Contains the list of the connected USB devices. Each string has the following format: "<ID1>:<ID2> <NAME>", for example "0000:1111 Company Ltd. Optical Mouse". See http://www.linux-usb.org/usb-ids.html for more about ID1 and ID2.
webCameras	Contains the map of the connected web cameras. Each entry has a camera device as a key, and the camera name as a value.
idleTimeout	Contains timeout in seconds. After that time of user inactivity idle() signal fires. Default value is 5.
envVariable()	Returns value of environment variable name or empty string if there is no a variable with that name.
setEnvVariable()	Adds the variable name to the environment with the value, if name does not already exist. If name does exist in the environment, then its value is changed to value if overwrite is true; if overwrite is false, then the value of name is not changed. Returns zero on success, or -1 on error.
unsetEnvVariable()	Deletes the variable name from the environment. If name does not exist in the environment, then the method succeeds, and the environment is unchanged. Returns zero on success, or -1 on error.
screenshot()	Takes screenshot and scales it to width by height size with respect of aspect ratio.
debug()	Sends debug message to syslog and to event system.
info()	Sends info message to syslog and to event system.
warning()	Sends warning message to syslog and to event system.
error()	Sends error message to syslog and to event system.
moveMouse()	Move the mouse cursor to the specified global position.
mouseClick()	Simulate the mouse button click. If button is 0 (the default) the left button is simulated, and the right otherwise.
mouseDoubleClicked()	Fires when some mouse button is double clicked.
mouseMoved()	Fires when mouse is moving.
mousePressed()	Fires when some mouse button is pressed.
mouseReleased()	Fires when some mouse button is released.
mouseWheelRotated()	Fires when mouse wheel is rotated.
idle()	Fires when system is in idle state (no user interaction).

usbConnected()	Fires when USB device is connected. The string parameter has the following format: "<ID1>:<ID2> <NAME>", for example "0000:1111 Company Ltd. Optical Mouse". See http://www.linux-usb.org/usb-ids.html for more about ID1 and ID2. Please remember that all the existing USB devices will fire this signal during application startup, so if you have 5 USB devices connected you will get 5 usbConnected() signals with ids of that devices during application startup.
usbDisconnected()	Fires when USB device is disconnected. The string parameter has the following format: "<ID1>:<ID2> <NAME>", for example "0000:1111 Company Ltd. Optical Mouse". See http://www.linux-usb.org/usb-ids.html for more about ID1 and ID2.

screenshot() returns object, which implements Screenshot interface:

```
interface Screenshot
    {
    readonly attribute int width; readonly
    attribute int height; readonly attribute string
    base64Data;
    }
```

The following HTML contains an example of global.system usage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>..:: global .system example ::..</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<s
tyl
e>
bo
dy
{
    margin: 32px;
    background-color: #EEEEEE;
    font-weight: bold;
    font-family: Arial;
    font-size: 20px;
    color: #333333;
}
</style>
<script type="text/javascript">

var
textVarName;
var textVarValue;
var screenshotImage;

function init ()
{
    textVarName = document.getElementById("textVarName"); textVarValue =
    document.getElementById("textVarValue"); screenshotImage =
    document.getElementById("screenshotImage");
    document.getElementById("screen").innerHTML = global.system.screen;
    document.getElementById("masterScreen").innerHTML = global.system.masterScreen;
}
```

```
function get()
{
	textVarValue.value = global.system.envVariable(textVarName.value);
}

function set ()
{
	global .system.setEnvVariable(textVarName.value, textVarValue.value);
}

function unset()
{
	global .system.unsetEnvVariable(textVarName.value);
}

function getScreenshot()
{
	global .system.screenshot(640, 480).assignToHTMLImageElement(screenshotImage);
}

</script>
</head>

<body onload="init()">
	<div style="text-align: center ;"> Screen:
		<span id="screen"></span>
	</div>
	<div style="text-align: center ;">
		Master screen: <span id="masterScreen"></span>
	</div>
	<div style="text-align: center ;">
		Environment variable name
		<input id="textVarName" type="text" /> Value
		<input id="textVarValue" type="text" />
		<input type="button" value="Get" onclick="get()" />
		<input type="button" value="Set" onclick="set()" />
		<input type="button" value="Unset" onclick="unset()" />
		</div>
		<div style="text-align: center ;">
		<img id="screenshotImage" align="middle" width="640" height="480" /><br>
		<input type="button" value="Get screenshot" onclick="getScreenshot()" />
	</div>
</body>
</html>
```

global.videoEncoder Object

The global.videoEncoder object allows the web application to take a video feed from either a webcam or a HDMI Source and encode it to MPEG2-TS (MPEG2 Transport Stream) and stream it out to an endpoint either via UDP or via TCP. This object can be coupled with the video player and can serve as a local view of the encoded frame that is being sent out on the wire.

Note While using TCP as the connection type, ensure that the TCP endpoint on the host to which you are interested to stream to is listening on the port of interest.

Note While using the webcam, make sure the Input resolution that you select is supported by the camera. It is recommended that you use either a 720p or a 1080i camera. A lower resolution camera may not give the desired output.

```
interface videoEncoder
{
        readonly attribute bool isAvailable; // Checks if Encoder is Available readonly attribute int status;
        readonly attribute int errorCode; readonly
        attribute string errorMessage;

        readonly attribute int videoInputCount;
        readonly attribute stringlist videoInputDescription; readonly
        attribute string snapshot;

        attribute string targetHost; // Target Host where MPEG2-TS has to be sent attribute int targetPort; // Target Port
        on Target Host to Receive it
        attribute int protocol;   // Udp=0, Tcp=1

        attribute int videoMode;   // SD=0, HD=1, CUSTOM=2
        attribute int videoSource; // Must be in [0, videoInputCount] range. 0 is for HDMI, 1-videoInputCount is for
    webcams

        attribute int h264Profile; attribute
        int inputResolution; attribute bool
        isProgressive; attribute int
        streamType;
        attribute int inputFrameRate; // 15, 24, 30, 60
        attribute int outputFrameRate; // 15, 24, 30, 60 attribute int
        averageOutputBitRate;
        attribute int minimumOutputBitRate;
        attribute int maximumOutputBitRate;
        attribute int outputResolution; attribute int
        audioBitRate;

signals:
        notready();
        ready();
        started();
        stopped();
        error(in int code, in string message);
slots:
        start();
        stop();
        takeSnapShot();

}
```

Table 4-17 global.videoEncoder Object Variables

Variable	Description
isAvailable	Use this routine to check if the box has videoEncoder Module in it. A 'true' value indicates presence of the module.

status	Represents status of video encoder. The status are listed in the section below as enumeration for your reference.
errorCode	This attribute will be set when an error occurs. Use this attribute for error handling in your application.
errorMessage	This routine returns the error string corresponding to the errorCode that was set. Use this function to display an error message to the user in your application.
videoInputCount	Returns the available video sources including the HDMI input from the USB dongle and all available webcams.
videoInputDescription	Returns the description of all video input devices present in the box.
snapshot	Returns the captured snapshot from the HDMI encode stream as base64data. The returned data is the JPEG image.
targetHost	Returns the target host to which the encoded stream is being sent.
targetPort	Returns the port number on which the encoded stream is being sent.
protocol	Returns integer value for the Transport Protocol: '0' for UDP or '1' for TCP.
videoMode	Returns video mode on which the video encoder is operating. Values are: '0' - SD, '1' - HD, and '2' for Custom.
videoSource	Returns the video source being selected for encoding either as '0' for HDMI or '1' for videoInputCount for webcam.
h264Profile	Returns the encoding H.264 profile being used by the encoder: '0' for Baseline, '1' for Main, and '2' for Extended Profile.
inputResolution	Returns the input resolution that is being used for the source. Possible values are '0' for 1920x1080 resolution, '1' for 1280x720, and '2' for 1024x600.
isProgressive	Returns 'true' if the scan format is set to Progressive.
streamType	Returns the input stream type that is configured on the encoder. Possible values are '0' = Program Stream, '1' = Transport Stream, '2' = Mpeg4 Stream (default), '3' = Elementary Stream, and '4' = Raw Stream.
inputFrameRate	Returns the Video-In Frame Rate as integer value in fps. Possible values are '0' for 15fps, '1' for 24fps, '2' for 30fps, and '3' for 60 fps.
outputFrameRate	Returns the Video-Out Frame Rate as integer value in fps. Possible values are '0' for 15fps, '1' for 24fps,

	'2' for 30fps, and '3' for 60 fps.
averageOutputBitRate	Returns the Video-Out Average Bit Rate in kbps
minimumOutputBitRate	Returns the Video-Out Minimum Bit Rate in kbps
maximumOutputBitRate	Returns the Video-Out Maximum Bit Rate in kbps
outputResolution	Returns the Video-Out Resolution from the encoding stream. Possible values are '0' for 1920x1080, '1' for 1280x720, '2' for 1200x672, '3' for 1168x656, '4' for 1024x576, and '5' for 768x432.
audioBitRate	Returns the Audio-Out Bit Rate being sent from the encoder in bps.
setTargetHost(in string targetHost)	Allows you to set the Target Host (IP Address either as Unicast or Multicast ipv4 Address).
setTargetPort(in string targetPort)	Allows you to set the Target Host's (Layer4) port Number (TCP or UDP port number).
setProtocol(in int transportProtocol)	Allows the Transport (Layer4) Protocol to be used when sending the encoded stream. Choices are '0' for UDP or '1' for TCP.
setVideoMode(in int videoMode)	Allows you to set video Encode mode either as SD (Standard Definition) or HD (High Definition). If you would like to still fine tune the encoding properties, you can select the custom option. Choices are '0' for SD, '1' for HD, and '2' for Custom.
setVideoSource(in int videoSource)	Allows you to set the video source for the encoder. Choices are '0' for HDMI Input from USB Dongle, and '1' for all available (v4l compliant) webcams.
setH264Profile(in int h264Profile)	Allows you to set the H.264 profile to be used for encoding. Choices are '0' for baseline profile, '1' for main profile, and '2' (default) for extended profile.
setInputResolution(in int inputResolution)	Allows you to set the input resolution for the video source. Choices are '0' for 1920x1080, '1' for 1280x720, and '2' for 1024x600.
setProgressive(in int flag)	Allows you to set the input scan format to Progressive. Call this API with parameter of '1' to set to Progressive. Choices are '0' or '1'.
setStreamType(in int streamType)	Allows you to set the stream type for the Video-In stream. Choices are '0' for PS, '1' for TS (default), '2' for Mp4, '3' for ES, and '4' for Raw.
setInputFrameRate(in int frameRate)	Allows you to set the Incoming (Video-In) Frame rate in fps. Choices are '0' for 15fps, '1' for 24fps, '2' for 30fps, and '3' for 60fps.

setOutputFrameRate(in int frameRate)	Allows you to set the Output (Video-Out) Frame rate in fps. Choices are '0' for 15fps, '1' for 24fps, '2' for 30fps, and '3' for 60fps.
setAverageOutputFrameRate(in int avgRate)	Allows you to set the Average Output Rate (Video-Out) in kbps.
setMinimumOutputBitRate(in int minRate)	Allows you to set the Minimum Output Rate (Video-Out) in kbps.
setMaximumOutputBitRate(in int maxRate)	Allows you to set the Maximum Output Rate (Video-Out) in kbps.
setOutputResolution(in int outputResolution)	Allows you to set the Output Resolution for Video-Out. Choices are '0' for 1920x1280, '1' for 1280x720, '2' for 1200x672, '3' for 1168x656, '4' for 1024x576, and '5' for 768x432.
setAudioBitRate(in int bitRate)	Allows you to set the Audio-In bit rate in kbps.

global.videoEncoder object Enumeration

```
{
        enum ErrorCodes
        {
            videoEncoderNotPresent = -1,
            UnabletoStopStreaming = -2,
            UnabletoStartStartStreaming = -3,
            RemoteSideNotListening = -4,
            MemoryExuastionError = -5,
            NoHdmiVideoSignal = -6,
            BothUsbAndPcieTogetherNotSupported = -7,
            HdmiVideoFormatNotUnderstood = -8,
        };
        enum Protocol
        {
            ProtocolUdp = 0,
            ProtocolTcp = 1,
        };
        enum VideoMode
        {
            VideoModeSD = 0,
            VideoModeHD = 1,
            VideoModeCustom = 2,
        };
        enum VideoSource
        {
            HdmiVideo = 0,
            WebcamVideo = 1,
        };
        enum H264Profile
        {
            H264ProfileBaseLine = 0,
            H264ProfileMain = 1,
            H264ProfileExtended = 2,
        };
        enum InputResolution
        {
            InputResolution1920x1080 = 0,
```

```
                InputResolution1280x720 = 1,
                InputResolution1024x600 = 2,
        };
        enum OutputResolution
        {
                OutputResolution1920x1080 = 0,
                OutputResolution1280x720   = 1,
                OutputResolution1200x672 = 2,
                OutputResolution1168x656 = 3,
                OutputResolution1024x576 = 4,
                OutputResolution768x432 = 5,
        };
        enum StreamType
        {
                StreamPgoram = 0,
                StreamTransport = 1,
                StreamMp4 = 2,
                StreamElementary = 3,
                StreamRaw = 4,
        };
        enum InputFrameRate
        {
                Input15fps = 0,
                Input24fps = 1,
                Input30fps = 2,
                Input60fps = 3,
        };
        enum OutputFrameRate
        {
                Output15fps = 0,
                Output24fps = 1,
                Output30fps = 2,
                Output60fps = 3,
    };
}
```

The following is an example of global.videoEncoder usage (file../../test/js/encoder.html):

```
<!doctype html>
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>..:: Encoder App ::..</title>
<style type="text/css">

html, body {width:100%; height:100%; padding:0; margin:0; position: absolute;}
* {-webkit-box-sizing:border-box; -moz-box-sizing:border-box; box-sizing:border-box;}

div, span, td, ul , ol , li , h1, h2, h3, h4, h5, h6, iframe, form, table , td, tr , th
{padding:0; margin:0; border:none} ul , ol ,
li { list -style:none;} img {border:none;}

@font-face {font-family: timerDisplay; src: url(fonts/PFDinTextPro-Regular.ttf)}

body {background: #777; color: #ffffff ; font :normal 12px Arial, Helvetica, sans-serif;} object {display:block; background:black}
.cameraDisabled, .cameraReady, .cameraOnline, .cameraError {width:44px; height:24px; background:transparent
url('img/encoder/LED.png') 0px 0px no-repeat;}
.cameraReady {background-position:-44px 0}
.cameraOnline {background-position:-88px 0}
```

```
.cameraError {background-position:-132px 0}

/*.status .devicetype .devicename {color:#848d9d; text-shadow:0 -1px 1px #000; font:normal 14px Arial, Helvetica, sans-serif;
padding:0 20px}
. calltime {color:#b1b6c3; font:normal 14px Arial, Helvetica, sans-serif ; height:41px; text-shadow:0 -1px 1px #000;}
.buttons {text-align:center; display: inline-block;}*/

/* ============= */
.topPanel {background:#0e1014 url('img/encoder/top_panel_background.png') left bottom repeat-x; color:#4c5058;
text-shadow:0 -1px 1px #000; height:64px; padding-bottom:7px; border-top:solid 1px #444444}
.bottomPanel {background:#0e1014 url('img/encoder/status_panel_background.png') top left repeat-x; color:#4c5058;
text-shadow:0 -1px 0 #000; padding:0 30px; text-align:center; font:normal 12px Arial, Helvetica, sans-serif; height:23px;
line-height:23px;
vertical-align:middle; overflow:hidden}

. display {border-style: solid ; border-width: 0px 10px 0px 10px;
-webkit-border-image:url('img/encoder/small_rec_display.png') 0 10 0 10 stretch;
border-image:url('img/encoder/small_rec_display.png') 0 10 0 10 fill stretch; height:35px; line-height:35px; width:100%;
text-align:center}
.timerOn, .timerOff {color:# ffffff ; font :normal 26px/35px timerDisplay, Arial, Helvetica, sans-serif ; text-
shadow:0 -1px 1px #000; vertical-align:middle; display:inline-block; margin-top:3px; overflow:hidden}
.timerOff {opacity:.1}
.startButton, .stopButton, .startButtonON {width:50px; height:50px; line-height:50px; padding:0; margin:0;
background:transparent url('img/encoder/buttons.png') 0px 0px no-repeat; border:none; text-shadow:0 -1px 0 black}
.startButton:hover, .stopButton:hover {background-position:0px 0px}
.startButton:active , .stopButton:active {background-position:-50px 0px}
.startButton:disabled, .stopButton:disabled {background-position:0px 0px}
.startButtonON {background-position:-100px 0px}
.startButtonON:hover {background-position:-100px 0px}
.startButtonON:active {background-position:-150px 0px}
.startButtonON:disabled {background-position:-100px 0px}
.startButton:active img, .stopButton:active img, .startButtonON:active img {opacity:.7}
.startButton:disabled img, .stopButton:disabled img {opacity:.4}
.view {background:#000; border:solid 1px #3b3d40; text-align:center; width:100%;}

/* hiding spin buttons */
input::-webkit-outer-spin-button, input::-
webkit-inner-spin-button {

/* display: none; <- Crashes Chrome on hover */
      -webkit-appearance: none;
      margin: 0; /* <-- Apparently some margin are still there even though it's hidden */
}
.sidepanel {background:#24252a url(img/encoder/sh.png) left top repeat-y; border-left:solid 1px
#777; max-width:400px;text-shadow:0 -1px 0 #000;}
#configWindow {display:block; text-align:center; width:400px; background:rgba(0,0,0,.8); color:#aaa; padding:20px;
border:solid 1px #333333; font:normal 12px Arial, Helvetica, sans-serif ; box-shadow:0 10px 30px rgba(0,0,0,.7)}.sidepanel
#configWindow {height:100%; box-shadow:none; background:none; border:none}
#configWindow fieldset {border:none; padding:20px}
#configWindow legend {text-align:center;}
#configWindow label {padding-right:20px; text-align:right; width:100%; display:inline-block}
#configWindow input[type=text], #configWindow input[type=number]
{background
•                                                                                    •
color:#FEFEFE}
#configWindow input.invalid {background-color:#FFA7B1}
#configWindow table {width:100%;}
#configWindow td {width:48%; text-align:left;}
#configWindow button { background:none; border-style: solid; border-width: 0px 10px 0px 10px; -webkit-border-
image:url('img/encoder/btn_idle.png') 0 10 0 10 stretch;
border-image:url('img/encoder/btn_idle.png') 0 10 0 10 fill stretch; height:24px; line-height:24px; font:normal
12px/22px Arial, Helvetica, sans-serif; color:#191d21; text-shadow:0 1px 0 rgba(255,255,255,.3);}
#topMenu button {background:transparent url(img/encoder/btn_idle.png) left top no-repeat; width:172px; height:24px;
```

```
border:none; }
#configWindow button:hover {-webkit-border-image:url('img/encoder/btn_hover.png') 0 10 0
10 stretch; border-image:url('img/encoder/btn_hover.png') 0 10 0 10 fill stretch;}
#configWindow button:active {-webkit-border-image:url('img/encoder/btn_pressed.png') 0 10
0 10 stretch; border-image:url('img/encoder/btn_pressed.png') 0 10 0 10 fill stretch; color:#002960}
#configWindow button:disabled {-webkit-border-image:url('img/encoder/btn_disabled.png') 0
10 0 10 stretch; border-image:url('img/encoder/btn_disabled.png') 0 10 0 10 fill stretch; color:#545b65; text-shadow:0 -1px 0
rgba(0,0,0,1);}
#configErrorMsg {background:#6b0202; color:#fff; text-align:center; margin:10px 30px; border:solid 1px red; border-
radius:4px; padding:6px}

/* debug */
.onSignal {margin-left:-13px; border-left:solid 3px white; padding-left:5px}
</style>
<script type="text/javascript"> var
player ;
var incall = false ; var
inputCount;
var Status;

var GUI_startButton, GUI_startButtonIcon, GUI_stopButton, GUI_stopButtonIcon, GUI_timer; var useApplicationData =
false;
var videoEncoder_targetHost; var
videoEncoder_targetPort; var
videoEncoder_videoSource; var
videoEncoder_videoMode; var
videoEncoder_protocol; var
encoderStatus;

var configWindow, targetAddr, targetPort, videoSource, videoMode, videoProtocol; function init (){
    player = document.getElementById("player"); GUI_startButton =
    document.getElementById("StartButton"); GUI_startButton.className =
    "startButton"; GUI_startButton.disabled = true;
    GUI_startButtonIcon = document.getElementById("StartButtonIcon");
    GUI_startButtonIcon.src = "img/encoder/start_icon.png"; GUI_stopButton =
    document.getElementById("StopButton"); GUI_stopButton.className = "startButton";
    GUI_stopButton.disabled = true;
    GUI_stopButtonIcon = document.getElementById("StopButtonIcon");
    GUI_stopButtonIcon.src = "img/encoder/stop_icon.png"; GUI_timer =
    document.getElementById("theTime");
    configWindow = document.getElementById("configWindow"); targetAddr =
    document.getElementById("targetAddr"); targetPort =
    document.getElementById("targetPort"); videoSource =
    document.getElementById("videoSource"); videoMode =
    document.getElementById("videoMode"); videoProtocol =
    document.getElementById("videoProtocol");
    // looking for 'useAppData' keyword in the query string var
    l=String(window.location);
    var qs=l.substring( l .indexOf("?", 0)+1, l .length) ; useApplicationData =
    (qs.indexOf("useAppData", 0)>=0);

    if (useApplicationData){
        readAppData(); //fill config form with values from appData.
        applyParams(false); // applying values (but not saving to registry ) . In case if invalid data the config panel with
        highlighted errors will appear.
    } else {
        //popup configuration window
        configWindow.style.display = "block";
    }
}

function readAppData(){
```

```
        targetAddr.value = global.applicationData.value("encoder.target", null ); targetPort.value =
        global.applicationData.value("encoder.port", null ) ; if (global
        .applicationData.contains("encoder.videoSource")){
                var val = global.applicationData.value("encoder.videoSource");
                for (var i=0; i<videoSource.length; i++){videoSource.options[ i ]. selected = (videoSource.options[i ].
                value.toUpperCase()==val.toUpperCase())
                }
        }
        if (global .applicationData.contains("encoder.videoMode")){
                var val = global.applicationData.value("encoder.videoMode");
                for (var i=0; i<videoMode.length; i++){videoMode.options[i]. selected =
                (videoMode.options[i].value.toUpperCase()==val.toUpperCase())
                }
        }
        if (global .applicationData.contains("encoder.protocol")){ var val =
        global.applicationData.value("encoder.protocol");
        for (var i=0; i<videoProtocol.length; i++){videoProtocol.options[ i ]. selected = (videoProtocol.options[i ].
        value.toUpperCase()==val.toUpperCase())
                }
        }
}


  function applyParams(save){
        var configErrorMsg = document.getElementById("configErrorMsg"); var invalid = false ;
        // Validating inputs
        if (targetAddr.value==null || targetAddr.value==""){ targetAddr.className = "invalid";
                invalid = true;
        } else {
                targetAddr.className = null;
        }
        if (isNaN(Number(targetPort.value)) || Number(targetPort.value)<257 || Number(targetPort.value)>65535){
                targetPort.className = "invalid"; invalid
                = true;
        } else {
                targetPort.className = null;
        }

        if (! invalid ){
                videoEncoder_targetHost = targetAddr.value; videoEncoder_targetPort =
                Number(targetPort.value); videoEncoder_videoSource = videoSource.value;
                videoEncoder_videoMode = videoMode.value; videoEncoder_protocol =
                videoProtocol.value;

                if (!! save){
                // save values to registry
                if (global .applicationData.isReadOnly){
                alert ("Application Data is read only. Configuration params will be lost after reboot.");
                }
                global .applicationData.insert ("encoder.target", videoEncoder_targetHost); global .applicationData.insert
                ("encoder.port", videoEncoder_targetPort);
                global .applicationData.insert ("encoder.videoSource", videoEncoder_videoSource); global
                .applicationData.insert ("encoder.videoMode", videoEncoder_videoMode); global .applicationData.insert
                ("encoder.protocol", videoEncoder_protocol);
                }

                configErrorMsg.style. visibility = "hidden";
                configWindow.style.display = "none"; initVideoEncoder();
        } else {
                configWindow.style.display = "block"; configErrorMsg.style.
                visibility = "visible";
        }
}
```

```
function initVideoEncoder(){
        inputCount = global.videoEncoder.inputCount;
global .videoEncoder.targetHost = videoEncoder_targetHost; global .videoEncoder.targetPort = videoEncoder_targetPort;
        global .videoEncoder.videoSource = videoEncoder_videoSource; global
        .videoEncoder.videoMode = videoEncoder_videoMode; global
        .videoEncoder.protocol = videoEncoder_protocol;

        // Connect Signals
        global .videoEncoder.ready.connect(onReady); global
        .videoEncoder.notready.connect(onNotReady); global
        .videoEncoder.started.connect(onStarted); global
        .videoEncoder.stopped.connect(onStopped); global
        .videoEncoder.error.connect(onError);


        // For UDP Stream Viewer
        player .endReached.connect(player.play); global
        .videoEncoder.start();
}

// For Encoder App
function onReady() {
        GUI_startButton.className = "startButton"; GUI_startButton.disabled
        = false; GUI_stopButton.disabled = false;


        document.getElementById("Status").innerHTML = "Ready" document.getElementById("cameraStatusLED").className =
        "cameraDisabled"; document.getElementById("theTime").className = "timerOff";
}

function onNotReady() { GUI_startButton.className =
        "startButton"; GUI_startButton.disabled = true;
        GUI_stopButton.disabled = true;


        document.getElementById("Status").innerHTML = "Not Ready"
        document.getElementById("cameraStatusLED").className = "cameraDisabled";
        document.getElementById("theTime").className = "timerOff";
        countDown(0);
}

function   onStarted()   { GUI_startButton.className =
        "startButtonON"; GUI_startButton.disabled = true;
        GUI_stopButton.disabled = false;


        document.getElementById("Status").innerHTML = "Encoding";
        document.getElementById("cameraStatusLED").className="cameraOnline"; player
        .play("udp://@127.0.0.1:"+videoEncoder_targetPort); countDown(1);
}

function onStopped() { GUI_startButton.className =
        "startButton"; GUI_startButton.disabled = false;
        GUI_stopButton.disabled = true;


        document.getElementById("Status").innerHTML = "Stoped";
        document.getElementById("cameraStatusLED").className="cameraDisabled"; countDown(0);
}

function onPaused() { GUI_startButton.className =
        "startButton"; GUI_startButton.disabled = false;
```

```
        GUI_stopButton.disabled = true;
        document.getElementById("Status").innerHTML = "Paused";
        document.getElementById("cameraStatusLED").className="cameraDisabled"; countDown(0);
}

function onError(code, explanation) { GUI_startButton.className =
        "startButton"; GUI_startButton.disabled = false;
        GUI_stopButton.disabled = false;

        var msg = code + ' : ' + explaination; alert (msg);
        document.getElementById("Status").innerHTML = "<span style='color:#ff6920;'>Error: " + explaination + " </span>";
        document.getElementById("camera").src = "img/encoder/webcam_disabled.png";
}

function startEncoder(){
        GUI_startButton.disabled = false;
        GUI_stopButton.disabled = false; global
        .videoEncoder.start();
}

function stopEncoder(){
GUI_startButton.disabled = false;
GUI_stopButton.disabled = false; global
.videoEncoder.stop();
}

// For Timing to be shown
var timer_sec = 00; // set the seconds var
timer_min = 00; // set the minutes var timer_hrs
= 00; // set the Hours var timer_OneSecond;

function countDown(flag){
        clearTimeout(timer_OneSecond); //preventing many timeouts creation when receiving multiple signals at the same time
        var calltime ;
        if ( flag ) {
                timer_sec++;
                if (timer_sec == 59) {
                timer_sec = 00;
                timer_min = timer_min + 1;
                }
                if (timer_min == 59) {
                timer_min = 00;
                timer_hrs = timer_hrs + 1;
                }
                if (timer_sec <= 9){ timer_sec =
                "0" + timer_sec;
                }
                calltime = (timer_hrs<1 ? "" : ((timer_hrs<=9 ? "0" + timer_hrs : timer_hrs) + ":")) + (timer_min<=9 &&
                timer_hrs>0 ? "0" + timer_min : timer_min) + ":" + timer_sec; GUI_timer.innerHTML = calltime;
                GUI_timer.title = "Last call duration: "+calltime; timer_OneSecond
                = setTimeout("countDown(1)", 1000);
        } else {
                GUI_timer.innerHTML = "0:00";
                clearTimeout(timer_OneSecond);
                timer_sec = 00;
                timer_min = 00;
                writeLog("<span style='color:yellow'>Timer stopped.</span>");
        }
}
</script>
</head>
```

```
 <body onload="init()">

<table border="0" cellpadding="0" cellspacing="0" width="100%" height="100%">
        <tr>
                <td align="center" style="border-right:solid 1px #444444">
                <table border="0" cellspacing="0" cellpadding="0" style="box-shadow:0 10px 30px rgba(0,0,0,.7)">
                <tr>
                <td class="topPanel">
                <table width="100%" border="0" cellspacing="0" cellpadding="0">
                <tr align="center">
                <td width="25%"><button id="StartButton" class="startButton" onClick="startEncoder()"><img
                id="StartButtonIcon" src="img/encoder/start_icon.png"></button><button id="StopButton" class="stopButton"
                onClick="stopEncoder()"><img id="StopButtonIcon" src="img/encoder/stop_icon.png"></button></td>
                <td><div class="display"><span class="timerOff"
                id="theTime">0:00</span></div></td>
                <td width="25%"><img id="cameraStatusLED" class="cameraDisabled" src="img/encoder/spacer.png"align="left"
                width="44" height="24" /></td>
                </tr>
                </table>
                </td>
                </tr>
                <tr>
                <td><!-- For UDP Stream Player -->
                <object id="player" type="application/x-qt-plugin" classid="videoplayer" width="720" height="480">
                <param name="url" value="udp://@127.0.0.1:31319"/>
                <param name="controlsVisible" value="false"/>
                </object>
                </td>
                </tr>
                <tr>
                <td class="bottomPanel"><span class="status" id="Status">status</span></td>
                </tr>
                </table>
                </td>
                <td width="1" class="sidepanel">
                <div id="configWindow" style="display:none">
                <fieldset>
                <legend>Encoder configuration</legend>
                <table border="0" cellspacing="0" cellpadding="0" align="center">
                <tr>
                <td><label for="targetAddr">Target Address:</label></td>
                <td><input id="targetAddr" type="text" placeholder="0.0.0.0" onChange="this.className=null" value="225.1.1.1"
                required></td>
                </tr>
                <tr>
                <td><label for="targetPort">Target port:</label></td>
                <td><input id="targetPort" type="number" min="257" max="65535"
                onChange="this.className=null"placeholder="31319" value="31319" required></td>
                </tr>
                <tr>
                <td><label for="videoSource">Video source:</label></td>
                <td>
                <select id="videoSource">
                <option value="1">Webcam</option>
                <option value="0" selected>HDMI</option>
                </select>
                </td>
                </tr>
                <tr>
                <td><label for="videoMode">Video mode:</label></td>
                <td><select id="videoMode">
                <option value="0">SD</option>
```

```
                <option value="1" selected>HD</option>
                <option value="2">Custom</option>
                </select></td>
                </tr>
                <tr>
                <td><label for="videoProtocol">Protocol:</label></td>
                <td><select id="videoProtocol">
                <option value="0" selected>UDP</option>
                <option value="1">TCP</option>
                </select></td>
                </tr>
                </table>
                <div id="configErrorMsg" style="visibility:hidden">Invalid params.<br>Please, fix values marked red.</div>
                <button onClick="applyParams(false)">Apply</button> <button
                onClick="applyParams(true)">Save and Apply</button> <button
                onClick="readAppData()">Use AppData</button>
                </fieldset>
                <div style="padding:20px 50px">
                Note: if application.data property is defined with a policy, configuration values cannot be saved on the
                management server and will be lost after reboot.
                </div>
                </div>
                </td>
        </tr>
</table>

<div style="display:none; visibility :hidden">
<!-- chaching images -->
<img src="img/encoder/start_icon.png" width="0" height="0" />
<img src="img/encoder/stop_icon.png" width="0" height="0" />
<img src="img/encoder/encoder/pause_icon.png" width="0" height="0" />
</div>

<!-- This part of code is used for tracing application states -->
<div id="appDebugInfo" style="visibility:hidden; background:rgba(0,0,0,.75); position:absolute; top:0; left :0;
width:300px;overflow-x:hidden; overflow-y: scroll ; color :white; padding:20px; font:normal 10px/12px sans-serif">
</div>
<script>
function writeLog(msg){
        var c = document.getElementById('appDebugInfo'); if (c){
                c.innerHTML=c.innerHTML + msg + "<br><br>";
        }
}
function isDebugMode(){
        var l=String(window.location);
        var qs=l.substring( l .indexOf("?", 0)+1, l .length) ; if
        (qs.indexOf("debug", 0)>=0){
                document.getElementById('appDebugInfo').style.visibility = "visible";
        } else {
                document.getElementById('appDebugInfo').style.visibility = "hidden";
        }
}
isDebugMode();
document.getElementById('appDebugInfo').style.height = document.body.offsetHeight+"px";
</script>

</body>
</html>
```

global.vncServer Object

The global.vncServer object allows you to control a local Virtual Network Computing (VNC) server. This server can be used to give virtual access to the IEC device such as to share a remote desktop. The remote desktop

requires a VNC client. You can establish a remote session from either direction (IEC or remote desktop) or from both directions.

Note This object will not traverse firewalls.

The global.vncServer object code is:

```
interface VncServer
{
    bool start(in string password = "", in string host = "", in int port = -1, in bool readOnly = false);
    bool start(in rect rc, in string password = "", in string host = "", in int port = -1, in bool readOnly = false);
    bool start(in int x, in int y, in int width, in int height, in string password ="", in string host = "", in int port = -1, in bool
    readOnly = false);

    readonly attribute bool isRunning; readonly
    attribute bool isReadOnly; readonly
    attribute Rect geometry;

slots:
    bool stop();
};
```

Table 4-18 global.vncserver Variables

Variable	Description
start(in string password, in string host, in int port, in boolreadOnly)	Starts the VNC server for full screen. If password is not empty, use this password to protect the VNC connection. If host is not empty, try to connect to this host, and start locally otherwise. If port is less than zero, use the default VNC port, or use this port otherwise. If readOnly is true, the VNC server does not accept any input from connected clients. Returns true in the case of success and false in the case of failure.
start(in int x, in int y, in int width, in int height, in string password, in string host, in int port, in bool readOnly	Starts the VNC server for rectangular area (x; y;width; height). If password is not empty, use this password to protect the VNC connection. If host is not empty, try to connect to this host, and start locally otherwise. If port is less than zero, use the default VNC port, or use this port otherwise. If readOnly is true, the VNC server does not accept any input from connected clients. If the rectangular area is empty, this method does nothing and returns false. Returns true in the case of success and false in the case of failure.

start(in rect rc, in string password, in string host, in int port, in bool readOnly)	Starts the VNC server for rectangular area (rc). If password is not empty, use this password to protect the VNC connection. If host is not empty, try to connect to this host, and start locally otherwise. If port is less than zero, use the default VNC port, or use this port otherwise. If readOnly is true, the VNC server does not accept any input from connected clients. If the rectangular area is empty, this method does nothing and returns false. Returns true in the case of success and false in the case of failure.
isRunning	Returns true if the VNC server is running.
isReadOnly	Returns true if the VNC server does not accept any input from connected VNC clients.
geometry	Returns geometry of the rectangular area the VNC server is running for. If the server is not running or no coordinates have been specified via start(), returns an empty rectangle.
stop()	Stops the VNC server. Returns true in the case of success (or the server is not running) and false in the case of failure.

Note You should avoid running the VNC server for areas containing video, either playing in Flash or in the VideoPlayer widget.

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">

<title>..:: global.vncserver test ::..</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<s

ty

le

>

b

o

dy
{
    margin: 20px;
    background-color:
    #111111; color: #eeeeee;
```

```
        font-weight:
        bold; font-family:
        Arial; font-size:
        18px;
}
input {background-color:#fffffe}
        .note {color:#91CAFF}
</style>

<script type="text/javascript">

var hostId;
var portId;
var roId;
var passwordId;
var xId, yId, widthId, heightId;

function isNumber(n)
{
        return !isNaN(n) && isFinite(n);
}

function init()
{
        hostId   =   document.getElementById("host");
        portId   =   document.getElementById("port");
        roId = document.getElementById("ro");
        passwordId = document.getElementById("password");

        xId = document.getElementById("x");
        yId = document.getElementById("y");
        widthId = document.getElementById("width");
        heightId = document.getElementById("height");
}

function start()
{
        try
        {
        // determine port
        var port = parseInt(portId.value);

        if(!isNumber(port))
                port = -1;

        // coordinates
        var x = parseInt(xId.value);
        var y = parseInt(yId.value);
        var width = parseInt(widthId.value); var
        height = parseInt(heightId.value);

        var useCoordinates = (isNumber(x) && isNumber(y) && isNumber(width) && isNumber(height));

        // read-only
        var ro = roId.checked;

        alert("Started: " + (useCoordinates ? global.vncserver.start(x, y, width, height, passwordId.value,
        hostId.value, port, ro) : global.vncserver.start(passwordId.value, hostId.value, port, ro)));
```

```
        }
        catch(ex)
        {
        alert("Exception: " + ex);
        }
}

function stop()
{
        try
        {
        alert("Stopped: " + global.vncserver.stop());
        }
        catch(ex)
        {
        alert("Exception: " + ex);
        }
}

</script>

</head>

<body onload="init()">
        <div class="note">Protect your connection with a password (leave empty to ignore):</div>
        Password: <input id="password" type="text"> <br><br>

        <div class="note">To connect remotely use the following host (leave empty to start locally):</div>
        Remote host: <input id="host" type="text"> <br><br>

        <div class="note">Connection port (local or remote, 5900 by default):</div> Port: <input
        id="port" type="number"   min="257" max="65535"> <br><br>

        <div class="note">To disallow user activity check the following button:</div>
        <input id="ro" type="checkbox"> Read-only <br><br>

        <div class="note">To share a rectangle use the following coordinates:</div> x: <input id="x"
        type="text">
        y: <input id="y" type="text"> <br> width:
        <input id="width" type="text">
        height: <input id="height" type="text"> <br><br>

        <div class="note">Start or stop the VNC server with the given parameters:</div>
        <button onclick="start()">Start</button>
        <button onclick="stop()">Stop</button><br><br>

        <div class="note">VNC server status:</div>
        <button onclick="alert(global.vncserver.isRunning)">Is running?</button>
        <button onclick="alert(global.vncserver.isReadOnly)">Is read-only?</button>
        <button onclick="alert(global.vncserver.geometry)">Geometry?</button>
</body>
</html>
```

global.window Object

The global.window API can be used to manipulate browser windows.

Note Enter the absolute path in global.window.open().

The following is the global.window object interface implementation:

```
interface Window {
    Object open(in string url, in
        bool isModal = true,
        in bool showDecorations = false); bool
    close(in Object window);
    bool close(in DomElement e);

signals:

    void closed(in string windowObjectName);
}
```

Table 4-19 global.window Variables

Variable	Description
open()	Opens a new browser window. The URL must be a fully qualified URL; relative URLs are not supported. If showDecorations is false both navigation bar and window title will be hidden.
close()	Closes a window previously opened with the open() method. You can pass to this method either an Object returned from the open() method or any DOM element. In the case of the DOM element, the window that the DOM element belongs to will be closed.
closed()	This method is fired when windows opened with the open() method are closed.

New Window Object Interface Implementation

When you open a new window with open() method, you will get a new window object in return. That object has the following interface:

```
interface NewWindow {
    attribute string objectName;
    attribute int x;
    attribute int left;
    attribute int y;
    attribute int top;
    attribute int width;
    attribute int height;
}
```

Table 4-20 NewWindow Variables

Variable	Description
objectName	Name of the object: This name will be passed to closed() signal of global.window object when some window is closed. It is an empty string by default. It can be any string. Duplicates are not checked.
x, left	Horizontal position of the window

y, top	Vertical position of the window
width	Window width: If you have window decorations visible in your new window or if you have the debug panel enabled there, you are not able to make your window smaller then the minimum size of those decorations.
height	Window height

The following HTML contains an example of global.window usage (file.../../test/js/global-window.html):

```
<html>
<head>
	<title>..:: global .window API test ::..</title>
<s
tyl
e>
bo
dy
{
	margin: 0;
}
</style>
<script language="JavaScript"> var
display;
var
lastWindow;
var nextId = 0;
function openWindow()
{
	lastWindow = global.window.open("test/js/global-window-content.html", false , true) ;
	lastWindow.objectName = "window-" + (++nextId);
	display.innerHTML = "Window '" + lastWindow.objectName + "' opened."; lastWindow.x = 100;
	lastWindow.y = 100;
	lastWindow.width = 600;
	lastWindow.height = 600;
	global .system.info("New window geometry is ("
	+ lastWindow.width
	+ 'x'
	+ lastWindow.height
	+ '+'
	+ lastWindow.x
	+ '+'
	+ lastWindow.y
	+ ").");
}
function closeWindow()
{
	global .window.close(lastWindow);
}
function onWindowClosed(windowName)
{
	display.innerHTML = "Window '" + windowName + "' closed.";
}
function init ()
{
	display = document.getElementById("display"); global
	.window.closed.connect(onWindowClosed);
```

```
}
</script>
<body onload="init()">
	<div id="display">Status</div>
	<a href="javascript:openWindow()">Open new window</a>
	<br />
	<a href="javascript:closeWindow()">Close last opened window</a>
</body>
</html>
```

The following is usage of global.window for new window content (file ../../test/js/global-window-content.html):

```
<html>
<head>
	<title>..:: global .window API test (new window) ::..</title>
<s
tyl
e>
bo
dy
{
	margin: 0;
}
</style>
<script language="JavaScript">
function closeWindow()
{
	global .window.close(document.documentElement);
}
</script>
<body>
	<a href="javascript:closeWindow()">Close this window</a>
</body>
</html>
```