An Introduction to
ROLE Interwidget Communication

Dominik Renzel
Chair of Computer Science 5 - Databases & Information Systems
RWTH Aachen University, Germany

This work by Dominik Renzel is licensed under a Creative Commons Attribution-ShareAlike 3.0 License.
Contents

1 Introduction 1
   1.1 What is Interwidget Communication? 1
   1.2 Can I see ROLE Interwidget Communication in action? 1
   1.3 What can I do with the ROLE Interwidget Communication Framework? 2
   1.4 How does ROLE Interwidget Communication work? 2
   1.5 What are ROLE Interwidget Communication Intents? 3
   1.6 What is the context for ROLE Interwidget Communication? 3
   1.7 Which prerequisites must be met for ROLE Interwidget Communication? 4

2 API Documentation 4
   2.1 iwc.Client 4
   2.2 iwc.Client.connect 4
   2.3 iwc.Client.disconnect 4
   2.4 iwc.Client.publish 4

3 Tutorials 5
   3.1 How to create a basic ROLE IWC-enabled widget 5
   3.2 How to test ROLE IWC-enabled widgets in the ROLE Sandbox 7

4 Resources 8
   4.1 Demo Widgets 8
   4.2 Tools 8

5 Acknowledgements 8
1 Introduction

In this guide you will learn how to develop widgets enabled with ROLE Interwidget Communication (IWC). Before we guide you through the development process, you should first understand what interwidget communication is and what you can use it for.

1.1 What is Interwidget Communication?

In ROLE we put a focus on real-time collaboration and communication among learners in the context of widget-based learning environments. In such environments, learners combine multiple widgets to support them in certain learning situations. For example for foreign language learning, learners would combine a widget for reading texts in the foreign language, a translator widget, and a vocabulary trainer where the learner can enter new terms and their translations.

As such, each of these widgets realizes a dedicated functionality, and the learner has to manually combine these functionalities for himself. In our example, if the learner wants to add a new item to his vocabulary trainer, he would have to mark the foreign term in the text reading widget, copy the term, paste it in the translator widget, and finally copy and paste both foreign term and translation in the vocabulary trainer widget. It would be more desirable to directly have the three widgets integrated with each other to take away the burden of manual interaction. Such an integration becomes realizable with communication between the widgets within the learner’s local environment ("single-user-single-browser interwidget communication").

Furthermore, multiple learners should be enabled to communicate and collaborate with each other. Therefore consider the extended scenario of multiple learners discussing a set of foreign language videos. Each of them has a video player widget in his learning environment. Now one of the learners proposes to watch a specific scene in one of the videos. In order to watch this scene, he has to open the video and navigate his video player to that scene, and all other learners have to do the same. In order to watch the scene together, they will all have to synchronize to press the play button at the same time. It would be more desirable, if only one of them could do the navigation in his video player widget, such that it is reflected in the video player widgets of all others in real-time, just as if they were in the same classroom sharing one TV set to watch the scene together. Such a form of collaboration becomes realizable with communication between widgets across multiple remotely connected environments ("multi-user-multi-browser interwidget communication").

1.2 Can I see ROLE Interwidget Communication in action?

Yes, you can.

- A video available at http://www.youtube.com/watch?v=MzAZNaLrZcU&t=5m07s demonstrates ROLE IWC in action.

- You can also try out ROLE IWC yourself following the tutorials in Section 3.
1.3 What can I do with the ROLE Interwidget Communication Framework?

In general, the ROLE Interwidget Communication Framework enables developers to create arbitrary widgets supporting both forms of single-user-single-browser and multi-user-multi-browser interwidget communication.

1.4 How does ROLE Interwidget Communication work?

ROLE Interwidget Communication works on a publish-subscribe basis. Any ROLE IWC-enabled widget first subscribes to a specific channel. Once subscribed to the channel, the widget can

- process messages received from the channel.
- publish messages to the channel to be forwarded to all subscribers.

With the help of the ROLE Interwidget Communication Framework, developers only need to focus on these two tasks. The framework transparently realizes the management of local and remote channels and message transport between widgets and remotely connected widget environments. Technically, local IWC is realized with HTML5 Web Messaging, remote IWC with the XMPP Publish-Subscribe extension protocol. Within the ROLE SDK Reference Implementation, each ROLE Space maintains such a channel accessible to all space members and widgets. Messages sent across widgets are called *intents*. Their format is inspired by Google Android intents.

![ROLE Real-time Communication & Collaboration Architecture](image)

Figure 1: ROLE Real-time Communication & Collaboration Architecture
1.5 What are ROLE Interwidget Communication Intents?

Intents represent the messages sent between widgets locally and remotely. Each message thereby represents the intent of the publishing widget to cause an effect in any of the receiving widgets suited for consuming such a type of intent. Intent messages are encoded in JSON. An example intent as well as descriptions of each field are given in the following. Mandatory fields are marked with an asterisk (*).

```javascript
var intent = {
    "component": "",
    "sender": "http://widget.org/sender.xml",
    "data": "http://data.org/some/data",
    "dataType": "text/xml",
    "action": "ACTION_UPDATE",
    "categories": ["category1", "category2"],
    "flags": ["PUBLISH_GLOBAL"],
    "extras": {"key1": "val1", "key2": 2}
}
```

- **component**(*) (String) - the component name of a specific recipient widget (explicit intent) or the empty string to indicate broadcasting (implicit intent)
- **sender** (String) - the component name of the sender widget. A value of the form `user@domain.tld/resource?sender=<component name>` indicates that the intent has been received from a remote environment operated by a user with the XMPP JID `user@domain.tld/resource`.
- **action** (String) - the action to be performed by receivers (e.g. `ACTION_UPDATE`)
- **data**(*) (String) - data in form of a URI (e.g. `http://example.org`)
- **dataType**(*) (String) - the data type in MIME notation (e.g. `text/html`)
- **categories** (Array) - categories of widgets to process the intent (e.g. `"editor"`)
- **flags** (Array) - flags controlling intent processing (e.g. `"PUBLISH_GLOBAL"`)
- **extras** (JSON Object) - auxiliary data (e.g. `{"examplekey": "examplevalue"}`)

1.6 What is the context for ROLE Interwidget Communication?

The ROLE Reference Implementation provides ROLE Interwidget Communication in the context of ROLE Spaces. Technically, each space maintains its own XMPP Pubsub channel for remote IWC. A connection to this channel is established automatically by the ROLE IWC Framework. Developers only need to focus on publishing outgoing and processing incoming events (see tutorial in Section 3).
1.7 Which prerequisites must be met for ROLE Interwidget Communication?

A set of real-time communication and collaboration features in the ROLE Reference Implementation (including remote IWC) are only possible in combination with a pre-configured XMPP server. The remote form of ROLE IWC and the multiuser chat in ROLE spaces require such a connection. The sandbox installation of the ROLE Reference Implementation at http://role-sandbox.eu is already configured in this way.

2 API Documentation

This section describes the functionality of this library in terms of API documentation for (objects and their) functions, parameters, returns, and callbacks. In Section 3, you find tutorials demonstrating the use of this library.

2.1 iwc.Client

iwc.Client: constructor

Creates a new instance of an iwc.Client object.

2.2 iwc.Client.connect

iwc.Client.connect: function (callback)

Connects the client to the current interwidget communication channel. After successful call, the client is able to publish and receive intents. Received intents are passed to the callback function.

Parameters

- callback: function(intent) - a callback function with one parameter triggered by the ROLE IWC framework on incoming intents.

2.3 iwc.Client.disconnect

iwc.Client.disconnect: function()

When connected, disconnects the client from the current interwidget communication channel.

2.4 iwc.Client.publish

iwc.Client.publish: function(intent)

When connected, publishes a widget to the local or remote interwidget communication channel.

Parameters

- intent - an interwidget communication intent (cf. Section 1.5).
3 Tutorials

This section includes a set of tutorials to get developers started with the usage of this library.

3.1 How to create a basic ROLE IWC-enabled widget

This tutorial provides instructions on how to equip a widget with ROLE Interwidget Communication and addresses developers who intend to use this library to implement ROLE IWC-enabled widgets. The following steps are necessary to create a ROLE IWC-enabled widget.

1) Create Widget XML

First create a new file (e.g. widget.xml) and fill in the code template below. The two script elements in the CDATA block are necessary imports for ROLE IWC.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs
    title="Your widget title"
    description="Your widget description"
    author="Your name"
    author\_email="your@email"
  ></ModulePrefs>
  <Content type="html">
    <![CDATA[
      <script src="http://open-app.googlecode.com/files/openapp.js"></script>
      <script src="http://dbis.rwth-aachen.de/gadgets/iwc/lib/iwc.js"></script>
      //add your code
    ]]>}
  </Content>
</Module>
```

2) Connect to IWC Channel

Now create an instance of `iwc.Client` and connect it to the current IWC channel as soon as the widget is fully loaded. Add the following code snippet to your widget code within the CDATA block.
var iwcClient;

function init(){
    iwcClient = new iwc.Client();

    var iwcCallback = function(intent){
        console.log("Received Intent:");
        console.log(intent);
    };

    iwcClient.connect(iwcCallback);
}

gadgets.util.registerOnLoadHandler(init);

1. Create a new instance of an IWC client as variable available in the scope of your widget.

2. Define a callback function iwcCallback(intent) for processing incoming intents.

3. Call the client’s connect function passing the callback you just defined to connect the widget to the current IWC channel.

3) Publish Intents

To publish an intent, create an intent object as described in Section 1.5 above, leaving the sender field empty. Then

```javascript
if (iwc.util.validateIntent(intent)) {
    iwcClient.publish(intent);
}
```

1. Create an intent object as described in Section 1.5 leaving the sender field empty. In particular decide if your intent should be broadcasted to remote peers (flag PUBLISH_GLOBAL) or within the user’s local widget environment (flag PUBLISH_LOCAL).

2. Optionally use utility function iwc.util.validateIntent to validate the intent to be published.

3. Publish the intent using function iwc.publish
3.2 How to test ROLE IWC-enabled widgets in the ROLE Sandbox

The ROLE Sandbox at http://role-sandbox.eu is fully configured SaaS installation of the ROLE SDK for users and developers to play around with ROLE technologies, among them ROLE IWC. To play with ROLE IWC follow the instructions below.

1) Point your browser to http://role-sandbox.eu.

2) Sign in with your Google Account.

3) Create a space of your choice (e.g. http://role-sandbox.eu/spaces/myspace).

4a) Add IWC-enabled widgets to the space via the ROLE Widgetstore.
   1. Click on +Widgetstore in the left sidebar to open the ROLE Widgetstore.
   2. In the Widget Store search for keyword iwc.
   3. Click the Select button of the widgets you want to try out.
   4. Click +<Widgetname> in the left sidebar under Widgets to finally add the selected widget to your space.

4b) Add IWC-enabled widgets by URL
   1. Click on +URL in the left sidebar.
   2. Enter the URL of the widget to add in the appearing dialog.
   3. Confirm the dialog to finally add the widget to your space.

A couple of simple demonstrational widgets are available. You can either find these widgets in the ROLE Widget Store or use the URLs linked in Section 4. Alternatively you can join the space http://role-sandbox.eu/spaces/iwc, which is already populated with a couple of demo widgets.

5) Collaborate with peers

Now you need another user to join the space and collaborate with you using ROLE IWC. This user should follow the steps below.

1. Point your browser to the URL of a previously created space (e.g. http://role-sandbox.eu/spaces/myspace).

2. Sign in with your Google Account.

3. Join the space by clicking join in the Members section of the left sidebar. At the current state it might be necessary to reload.

4. Experience real-time collaboration by using the widgets
4 Resources

4.1 Demo Widgets

In this section we list a set of simple example widgets demonstrating the use of ROLE IWC for realtime collaboration among users. Every URL points to the relevant entry in the ROLE Widget Store, including widget URL.

- Collaborative Google Maps Navigation - a simple widget demonstrating remote collaboration via ROLE IWC. Users zoom/pan on a Google Maps canvas, their interactions with the map are synchronized in remote users’ environments.

- Collaborative Painting - a simple widget demonstrating remote collaboration via ROLE IWC. Users select painting tools (line width and color) and paint on a shared canvas, their interactions are synchronized in remote environments. Users can additionally save their work as PNG.

- Collaborative Youtube Video Navigation - a simple widget demonstrating how to watch and navigate videos together synchronously.

4.2 Tools

In this section we list a set of tool widgets supporting the development of ROLE IWC-enabled widgets. Every URL points to the relevant entry in the ROLE Widget Store, including widget URL.

- ROLE IWC Widget - a minimal example of a ROLE IWC-enabled widget; usable as template for developers.

- ROLE IWC Test Stub - a widget for tracing and publishing ROLE IWC intents.

5 Acknowledgements

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no 231396 (ROLE project).