

# **IM Ajax Support Component Specification**

## **1. Design**

IM is an application which allows users to perform online chat through the web browser. AJAX technology is used heavily to exchange messages in order to enhance user's experience. The IM Ajax Support component provides the servlets to handle the Ajax requests and responses on the server side.

The component will provide servlet support for the application on AJAX interactions. The input and output will be using XML. The interactions will be covered in different pages of the application. On the request side, Javascript will be used to compose the XML input and pass it to the servlet for processing. On the respond side, the output XML encapsulating the updated information/result will be parsed asynchronously using Javascript.

The main workflow is described as follow:

The javascript on client side composes a xml request, and sends it to the url, IMAjaxSupportServlet will first use RequestHandlerManager to pick up the matching handler to service the request. If the request message is invalid, or no matching handler is found, the manager will return error message to the client. If the handler is found, the manager will delegate to the handler to process. The handler will also return success or failure xml message based on the handling result. The request and response xml message must conform to the XSD schema files in docs/ directory. Please refer to the algorithm for the format of the request and response.

### **1.1 Design Patterns**

The RequestHandler implements the strategy pattern for RequestHandlerManager.

### **1.2 Industry Standards**

Servlet 2.4

XML

### **1.3 Required Algorithms**

#### **1.3.1 *The XMLrequest message***

In the version 1.0, there are seven concrete implementations for RequestHandler, and there are seven XML schemas for these handlers. All the schemas are given in the docs/ directory. The incoming http request message must confirm to one of those schemas. Otherwise, the client will always receive an error response, indicating that the request message is invalid or the message type is unrecognized. As the structure of the request message is very simple, this algorithm does not discuss them.

#### **1.3.2 *The XML response message***

All the RequestHandler implementations will set the xml response on http response no matter the operation succeeds or not. The content type for the http response is "text/xml". The XML response must follow the below schema:

```
<response>
```

```
<success>success message</success>
```

```

    <failure> failure message </failure>

    <messages>

        <message>...</message>

    </messages>

    <client_position>1</client_position>

</response>

```

The response element must contain one of success and failure element, but not both. If the success is included, the response indicates that the operation succeeded, otherwise, failed. The response element optional contains the messages element. The messages element must contain at least one message element which is defined in the IM Sales Messenger component. The response element optional contains the client\_position element. The node value of the client\_position must be integer or long.

Please see the XSD schema in the docs/response.xsd

### 1.3.2 *Generate XML response*

All the subclass of AbstractRequestHandler will generate the xml response. If the handler succeeded, and need to populate the messages element, the handler must first create a DateFormatContext, get the cookies from the http request, select the cookie whose name is IMAjaxSupportUtility#getTimezoneCookieName(), get the cookie value as the time zone id, and set the id on the DateFormatContext. Then, pass the DateFormatContext to the XMLMessage to obtain the xml string. Please note that if no matching cookie is found, simply pass the DateFormatContext to the message.

If the handler failed, the response only contains the failure element. The value of the failure element will be the exception message, or any other meaningful string indicating the error. For example, if the handler can not find the category in the http session, the error message will look like "no category found in the session". If user profile in the http session has wrong role, the error message will look like "Incorrect role name for the user profile"

### 1.3.3 *Log for the request handler*

All the subclass of AbstractRequestHandler will log the information at the Level.INFO level. Logging information must contain user ID, timestamp, action taken, and entity IDs affected. The user id is retrieved from the ChatUserProfile in http session.

The action name is read from the type attribute of the xml request element

The affected entity IDs are other ID involved in the handler, except the user ID.

The log format will be:

User ID:[user id] timestamp:[timestamp] action:[actionName]

affected entityIDs: [..]

For the affected ids, all the involved ids should be logged. For example, the AcceptChatRequestHandler is the most complicated one, this handler will use category id, session id and request user id. The format for the affected entityIDs should be:

affected entityIDs: requestUserId [userId] sessionId [sessionId] category ids [idOne] [idTwo]....

## 1.4 Component Class Overview

### **IMAjaxSupportServlet:**

IMAjaxSupportServlet provides servlet support for the application on AJAX interactions. It receives the client request and output the XML response to indicate if the operation succeeds or not.

Servlet container can guarantee the thread safety of this class.

#### **IMAjaxSupportUtility:**

IMAjaxSupportUtility holds the global configurable values (like session key, object key, etc), shared by all the other class in this component.

This class is thread safe since it only reads the values from the config manager..

#### **RequestHandlerManager:**

RequestHandlerManager is used to route the incoming request to the RequestHandler based on the request message type. The request handlers are loaded from the configuration file. If the request message is invalid, or no matching handler is found for the request, this manager will set the failure message on the http response and return.

This class is thread safe since it does not contain any mutable status..

#### **RequestHandler:**

RequestHandler defines the contract of handling the request based on some logic. Please be noted that, the implementation should not throw any exception. The exception should be caught, and the exception message should be contained as the part of the response.

The implementation is required to be thread safe.

#### **AbstractRequestHandler:**

AbstractRequestHandler provides an additional method to return the log for all the subclass that need to log something.

This class is thread safe since it's immutable.

#### **ReadClientUserMessageHandler:**

This handler is only for users with client role. The handler will pull the pending messages from the user message pool. In addition, the current position of the client (under the selected category) in the service engine is included in the response. The request message for this handler is defined in docs/ReadClientUserMessage.xsd. The response schema is given in the algorithm

This class is thread safe since it's immutable.

#### **ReadManagerUserMessageHandler:**

This handler is only for users with manager role. The handler will pull the pending messages from the user message pool.

The request message for this handler is defined in docs/ReadManagerUserMessage.xsd. The response schema is given in the algorithm

This class is thread safe since it's immutable.

#### **ReadClientSessionMessageHandler:**

This handler is only for users with client role. The xml request message includes the session id. The handler will pull the pending messages from both the user message pool and session message pool.

The request message for this handler is defined in docs/ReadClientSessionMessage.xsd. The response schema is given in the algorithm

This class is thread safe since it's immutable.

**ReadManagerSessionMessageHandler:**

This handler is only for users with manager role. The xml request message includes the session id. The servlet will pull the pending messages from the session message pool of the user.

The request message for this handler is defined in docs/ReadManagerSessionMessage.xsd. The response schema is given in the algorithm. This class is thread safe since it's immutable.

**AcceptChatRequestHandler:**

The xml request message includes the request user id and session id. The handler will update the state in the service engine to servicing for that session. The response will indicate whether the operation succeeds or not.

The request message for this handler is defined in docs/AcceptChatRequestHandler.xsd.

The response schema is given in the algorithm.

This class is thread safe since it's immutable.

**PostTextMessageHandler:**

The xml request message includes the session id and text. The handler will post the submitted message to all users of the session. In addition, the handler will pull the pending messages from the session message pool of the user and return them in the response.

The request message for this handler is defined in docs/PostTextMessageHandler.xsd.

The response schema is given in the algorithm.

This class is thread safe since it's immutable.

**ChangeManagerStatusHandler:**

This handler is only for users with manager role. The request parameter includes the status. The handler will change the user status. If it is ONLINE, the handler will enqueue the manager to the service engine for each of the selected categories. Conversely, if it is BUSY, the handler will dequeue the manager from the service engine.

The request message for this handler is defined in docs/ChangeManagerStatusHandler.xsd. The response schema is given in the algorithm.

This class is thread safe since it's immutable.

## 1.5 Component Exception Definitions

**IllegalArgumentException**

This exception is thrown in various methods if null object is not allowed, or the given string argument is empty. Refer to the documentation in Poseidon for more details.

**NOTE: Empty string means string of zero length or string full of white spaces.**

**IOException**

This exception is thrown by IMAjaxSupportServlet if any I/O error occurred.

**ServletException**

This exception is thrown by IMAjaxSupportServlet if any servlet-related error occurred.

**IMAjaxConfigurationException**

This exception will be thrown by IMAjaxSupportUtility if any error occurred during loading

the configuration values.

## **1.6 Thread Safety**

This component will be used in the web application, and must be thread safe. IMAjaxSupportServlet is not immutable, but servlet container can guarantee the thread safety of this class. RequestHandlerManager is thread safe since it's immutable. RequestHandler and its implementation are also required to be thread safe. IMAjaxSupportUtility is thread safe, as this class only reads the configuration values from the ConfigManager.

## **2. Environment Requirements**

### **2.1 Environment**

Java 1.4 or higher.

### **2.2 TopCoder Software Components**

#### **Base Exception 1.0**

The custom exception extends BaseException in this component.

#### **Sale IM Messenger 1.0**

It will be used to route the message.

#### **Chat Message Pool 1.1**

It's used to pull the message from the user pool or session pool

#### **Chat Session Manager 1.0**

ChatSession come from this component.

#### **Chat User Profile 2.0**

ChatUserProfile class comes from this component

#### **Logging Wrapper 1.2**

It's used to log the actions.

#### **Service Engine 1.0**

It's used to service the user request.

#### **IM Application Logic 1.0**

Responser and Requester objects come from this component.

#### **Chat Status Tracker 1.0**

It's used to update the manager status.

#### **Object Factory 2.0.1**

It's used to create object from the configuration file.

#### **Status Tracker 1.0**

The status class comes from this component.

#### **Configuration Manager 2.1.5**

It will be used to load the configuration values.

## 2.3 Third Party Components

None.

## 3. Installation and Configuration

### 3.1 Package Name

com.cronos.im.ajax

com.cronos.im.ajax.handlerimpl

### 3.2 Configuration Parameters

Configuration values for IMAjaxSuportUtility

Parameter	Description	Values
<b>client_role_name</b>	The client role name. <b>Required.</b>	client
<b>manager_role_name</b>	The manager role name <b>Required.</b>	manager
<b>role_property_name</b>	The property name used to get the role name from the user profile. <b>Required.</b>	role
<b>user_profile_session_key</b>	The key used to get the ChatUserProfile from the http session attribute. <b>Required.</b>	user_profile_session_key
<b>messenger_key</b>	The object key used to create the Messenger from object factory <b>Required.</b>	messenger_key
<b>chat_session_manager_key</b>	The object key used to create the ChatSessionManager from object factory <b>Required.</b>	chat_session_manager_key
<b>chat_status_tracker_key</b>	The object key used to create the ChatStatusTracker from object factory <b>Required.</b>	chat_status_tracker_key
<b>service_engine_key</b>	The object key used to create the ServiceEngine from object factory <b>Required.</b>	service_engine_key
<b>time_zone_cookie_name</b>	The cookie name used to get the time zone. <b>Required.</b>	timeZone
<b>handler_manager_key</b>	The object key used to create the HandlerManager from object factory. <b>Required</b>	handler_manager_key
<b>category_session_name</b>	The property name used to get the categories from http session attribute. <b>Required.</b>	category_session_name

Configuration values for RequestHandlerManager

Parameter	Description	Values
<b>handler_types</b>	Contains the type values. <b>Required.</b>	ReadClientUserMessage;ReadManagerUse rMessage;PostTextM essage
<b>xxx</b>	The "xxx" property name is one of the type values defined in "handler_types" property. The value of the xxx property is object key used to create the handler instance from object factory. <b>Required.</b>	read_client_user_me ssage_handler_key

### 3.3 Dependencies Configuration

The dependent components should be properly configured to make this component work.

## 4. Usage Notes

### 4.1 Required steps to test the component

- Extract the component distribution.
- Follow [Dependencies Configuration](#).
- Execute 'ant test' within the directory that the distribution was extracted to.

### 4.2 Required steps to use the component

Preload the configuration file into Configuration Manager. Follow demo.

### 4.3 Demo

The component will be deployed in the web application. A sample web.xml is given in docs. Assume that the url access the IMAjaxSupportServlet is <http://localhost:8080/im/IMAjaxSupport>.

The javascript on client side composes a xml request, and sends it to the url, IMAjaxSupportServlet will first use RequestHandlerManager to pick up the matching handler to service the request. If the request message is invalid, or no matching handler is found, the manager will return error message to the client. If the handler is found, the manager will delegate to the handler to process. The handler will also return success or failure xml message based on the handling result. The request and response xml message must conform to the XSD schema files in docs/ directory. Please refer to the algorithm for the format of the request and response.

The response for the ReadClientUserMessage is most complicated. The demo only shows the request and response for this message. Developers are required to show all the seven messages in the unit test.

Assume that the client sends the message:

```
<request type=" ReadClientUserMessage">
```

```
<session_id>1</session_id>
</request>
```

If the request is handled successfully, the response will contains the user messages and the current position of the client in the service engine.

```
<response>
  <success>success</success>
  <messages>
    <!--please see the Sales IM Messenger for the message schema-->
    <message type="someType">...</message>
    <message type="someType">...</message>
  </messages>
  <client_position>1</client_position>
</response>
```

## 5. Future Enhancements

None