RTView Advanced Design Topics
- Customization of Functions & Commands

Customization

- **View Manager** – custom class MyRtViewManager allows user to customize user views, including catching active/deactivate/update for each panel and rtv file, dynamically populating objects, dynamically making data attachments, custom tree view, custom tabbed views.

- **Function Handler** – custom class MyFunctionHandler allows implementation of a custom function, including both scalar and tabular data.

- **Popup Menus** – custom class MyJPopUpMenu allows implementation of custom pop-up menus, including pop-ups that are custom based on what panel and rtv file you are in, and what object you are over.

- **Message Handler** – custom class MyMessageHandler allows implementation of custom message handling, including making single-click do a double-click, doing multiple drill-downs, and responding to messages from pop-up menus.

- **Role Manager Handler** – custom classes MyUserManager and MyRoleManager are used to retrieve login information from LDAP or other access control mechanisms.

Customization (cont)

- **Command Handler** – custom class MyCommandHandler allows implementation of custom commands, which are accessed from command object property.

- **Data Sources** – custom class MyDS allows implementation of custom data sources, expanding the list of standard data sources currently available.

- **SL-GMS Objects** – custom GMSDraw objects can be used within the application, and be configured to respond graphically to changes in data values.

Customization - Functions

To implement a custom function, the following steps must be performed:

1. Create the custom class MyFunctionHandler.java source code file.

2. Implement the function within the java code.

3. Implement the return result method, as supported by the function.

   For example, implement the getTabularResult() method to return a table of values.

4. Implement the getFunctionDescriptors method for each of the supported return result methods.

5. Compile the class, and include in the myclasses.jar file. Upon compilation, the function's name will appear in the list of function types, at the bottom.

Customization - Functions

The following is the function specification:

```
public Vector getFunctionDescriptors ()
{
    Vector v = new Vector();

    v.addElement(   new GmsRtViewFunctionDescriptor("Long Concat",
                    new String[] {"Value 1", "Value 2”},
                    new String[] {"s_arg1", "s_arg2"},
                    GMS.G_STRING, null,
                    "This function returns a string", false));

    v.addElement(   new GmsRtViewFunctionDescriptor("Long Double Add",
                    new String[] {"Value 1", "Value 2"},
                    new String[] {"arg1", "arg2"},
                    GMS.G_DOUBLE, null,
                    "This function returns double sum", false));
    return v;
}
```

Customization - Functions

Example return result method implemented for returning a string datatype:

```
public String getStringResult (String functionName,
                                        GmsModelVariables functionIcon)
{
    System.out.println("getStringResult: " + functionName);

    String returnString = functionIcon.getGmsStringVar("s_arg1");
    returnString += functionIcon.getGmsStringVar("s_arg2");

    return returnString;
}
```

Customization - Functions

Limitations on Functions

Limit of 4 GmsTabularData arguments and 10 each of ints, doubles and Strings. In addition to this limitation, the edit function dialog also has a limitation of only allowing 10 total arguments.

The available arguments are:

GmsTabularDatas:	argTableColumn, argTableColumn2, argTableColumn3, argTableColumn4
Strings:	s_arg1, s_arg2, …, s_arg10
Integers:	i_arg1, i_arg2, …, i_arg10
Doubles:	arg1, arg2, …, arg10