DI2E-F QRC/JC2-CUI/GCCS-I3

Common Map Widget API

Version 1.0.0

Dated

1/19/2012

The intent of the Common Map Widget Application Program Interface (API) is to provide a common API for displaying and manipulating data on a map when the widget containing the map may be created by a different program than the ones creating the data widgets.

Common Map Widget API

Contents

Introduction

Background

Many programs and projects create widgets that search for or manipulate data then present the results on a map. The desire is to be able to combine data search/manipulation widgets from any provider with map widgets from other providers. In order to accomplish this, a standard way for the data search/manipulation widgets to be able to communicate with the map widget is necessary. This Application Program Interface (API) is the codification of that standard.

Overview

Using this API allows developers to focus on the problem domain rather than implementing a map widget themselves. It also allows the actual map implementation used to be chosen dynamically by the user at runtime rather than being chosen by the developer. Any map implementation that applies this API can be used. Currently, implementations using Google Earth, Google Maps V2, Google Maps V3, and OpenLayers APIs are available, and others can be written as needed.

Another benefit of this API is that it allows multiple widgets to collaboratively display data on a single map widget rather than forcing the user to have a separate map for each widget, so the user does not have to struggle with a different map user interface for each widget.

The API uses the OZONE Widget Framework (OWF) inter-widget communication mechanism to allow client widgets to interact with the map. Messages are sent to the appropriate channels (defined below), and the map updates its state accordingly. Other widgets interested in knowing the current map state can subscribe to these messages as well.

It is worth noting that the map itself may publish data to these channels on occasion. For example, a map.feature.selected message may originate from a widget asking that a particular feature be selected or because a user has selected the feature on the map.

While in most instances the map will not echo back another message to confirm that it has performed an operation, the map will send a view status message whenever the map view (zoom/pan) has been changed, either directly by the user or due to a view change message sent by another widget. This allows non-map widgets that wish to be aware of the current viewport to have that information without having to process all the various messages that can change its state.

In addition, it is expected that late-arriving widgets may wish to query the map for the current state. The API, therefore, includes support for widgets to request the current status so they can configure themselves appropriately.

How to use the channels and messages

There are two ways that the map in the map widget can be manipulated. The first is by other widgets via the Common Map Widget API. The second is by a user directly manipulating the map via the map Graphical User Interface (GUI) (for example, using a drawing tool on the map).

The first way: To manipulate the map, widgets send messages in the map channels, and the map widget responds by modifying its current state. Anyone can modify the current state, a data source widget might request that some Keyhole Markup Language (KML) be loaded, or that the view be changed to some other location. The map, *and any other widget configured to*, reacts to these change requests and modifies its state accordingly.

The second way: When changes are made to the map directly by the user (for example, the user uses a drawing tool to manipulate the map), the map will post messages to the map channels as though the map was being manipulated via another widget. This is so any widget listening to the map channels can respond the changes.

In other words, if a widget requests that a particular feature be selected, or the user selects a feature on the map, a map.feature.selected message is sent out to any widget that cares. The receiving widgets do not care if the selection originated from a user click on the map or another widget programmatically requested it.

Overlays

By default, all data added by an individual widget will be placed into a single overlay. By default, therefore, there will be one overlay per sending widget. The API also supports the ability to create and specify particular overlays when adding data. This allows a single widget to place data in multiple overlays as well as multiple widgets to place data in the same overlay. To prevent unintended merging of data due to unintended usage by multiple widgets of same overlay id, it is suggested that developers use the widget id as (at least part of) overlay ids if no sharing is intended and follow the guidelines recommended by the OWF documentation regarding preference namespaces for shared overlays to avoid unintended collisions.

Features and Feature Id's

Features in the context of this document refer to the discrete pieces of data passed to the API. A feature may be a single marker or polygon or be a complex feature (for example, a KML Document), which itself contains many sub-features. The feature id used by the API always refers to the feature id given when plotting the entire feature. The only time a sub-feature id is ever used is in the map.feature.selected message. In this message, the selectedId will contain the id of the lowest level feature selected (if available).

Payloads

All Payloads can be either a single object or an array of like objects.

Errors

Any message sent that is missing a required attribute should result in the map widget publishing an error message on the error channel. If the map widget is unable to find an object based on the given identifier, an error will be published as well. In general, any time the map is unable to complete a requested operation, an error will be published (if possible).

API

The channels associated with the Common Map Widget API are grouped according to the following:

map.overlay channels – Messages associated with creating and manipulating overlays.

map.feature channels – Messages associated with loading feature data onto the map.

map.view channels – Messages associated with manipulating the view.

map.status channels – Messages associated with obtaining the current map state.

map.error channels – Error messages.

map.overlay channels

Create Overlay

Purpose: Create an overlay into which data can be aggregated.

Channel: `map.overlay.create`

Payload: `{name: (optional), overlayId: (optional), parentId: (optional)}`

name: The name of the overlay. If not included, the id is used as the name. Note that overlay names do not have to be unique and are intended for display purposes only.

overlayId: The unique id of the new overlay. If no overlayId is included, default overlay with id equal to sending widget's id is assumed. If an overlay with the given id already exists, this message will have no effect. Note that overlay ids must be unique even across multiple parent overlays.

parentId: The id of the parent overlay in which to create this overlay.

Example:

```
{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data", "name": "Public SAGE data"}
```

Remove Overlay

Purpose: Remove entire overlay from the map.

Channel: `map.overlay.remove`

Payload: `{overlayId: (optional)}`

overlayId: The id of the overlay to be removed. If no overlayId is included, default overlay with id equal to sending widget's id is assumed.

Example: `{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-dd8c75434cf2.SAGE Public Access Data (no SAGE login required).Public SAGE data"}`

Hide Overlay

Purpose: Hide existing overlay on the map.

Channel: `map.overlay.hide`

Payload: `{overlayId: (optional)}`

overlayId: The id of the overlay to be hidden. If no overlayId is included, default overlay with id equal to sending widget's id is assumed.

Example: `{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-dd8c75434cf2.SAGE Public Access Data (no SAGE login required).Public SAGE data"}`

Show Overlay

Purpose: Show existing overlay on the map.

Channel: `map.overlay.show`

Payload: `{overlayId: (optional)}`

overlayId: The id of the overlay to be shown. If no overlayId is included default overlay with id equal to sending widget's id is assumed.

Example: `{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-dd8c75434cf2.SAGE Public Access Data (no SAGE login required).Public SAGE data"}`

map.feature channels

Plot Feature

Purpose: Have the map plot feature data.

Channel: `map.feature.plot`

Payload: `{overlayId: (optional), featureId: (required), name: (optional), format: (optional), feature: (required), zoom: (optional)}`

overlayId: The id of the overlay this feature should be loaded into. If overlay with this id already exists, new feature is merged into existing overlay; otherwise, new overlay will be created. If no overlayId is included, default overlay with id equal to sending widget's id is used. If overlay exists, it will retain its status (whether visible or hidden). If overlay is created, it will be made visible.

featureId: Unique identifier for the given feature data. Note that feature ids must be unique within a given overlay. Reusing a feature id will be considered a reload, with the original feature data being removed and replaced by the new feature data.

name: Name for the given feature data. Note that feature names do not have to be unique and are intended for display purposes only.

format: Data format of the given feature. If no format is specified, the format defaults to "kml." A list of supported formats supported by a particular map implementation may be obtained by querying the map using the map.status channel (see map.status). Note that for this version of the Common Map Widget API, the only format that all map implementations *must* support is KML.

feature: Feature data to be loaded into the map.

zoom: *true* if map should zoom to newly loaded feature data, *false* if not. Default is false.

Example:

```
{"overlayId": "User drawn", "name": "World Trade Center",
"featureId": "2d882141-0d9e-59d4-20bb-58e6d0460699.1", "feature":
"<kml xmlns=\"http://www.opengis.net/kml/2.2\"
xmlns:gx=\"http://www.google.com/kml/ext/2.2\"
xmlns:kml=\"http://www.opengis.net/kml/2.2\"
xmlns:atom=\"http://www.w3.org/2005/Atom\"><Placemark
id=\"2d882141-0d9e-59d4-20bb-58e6d0460699.1\"><name>World Trade
Center</name><description>Site of World Trade
Center.</description><Style><IconStyle><Icon><href>https://localh
ost/mapWidget2/images/blu-circle.png</href></Icon><hotSpot
x=\"0.5\" y=\"0\" xunits=\"fraction\"
yunits=\"fraction\"></hotSpot></IconStyle></Style><Point><coordin
ates>-74.01326179504395,40.71153538591555,0
</coordinates></Point></Placemark></kml>"}
```

Plot URL

Purpose: Have the map plot feature data from a Uniform Resource Locator (URL).

Channel: `map.feature.plot.url`

Payload:

```
{overlayId: (optional), featureId: (required), featureName:
(optional), format: (optional), url: (required), zoom:
(optional)}
```

overlayId: The id of the overlay this feature should be loaded into. If overlay with this id already exists, new feature is merged into existing overlay; otherwise, new overlay will be created. If no overlayId is included, default overlay with id equal to sending widget's id is used. If overlay exists, it will retain its status (whether visible or hidden). If overlay is created, it will be made visible.

featureId:	Unique identifier for the given feature data. Note that feature ids must be unique within a given overlay. Reusing a feature id will be considered a reload, with the original feature data being removed and replaced by the new feature data.
featureName:	Name for the given feature data. Note that feature names do not have to be unique and are intended for display purposes only.
format:	Data format of the given feature. If no format is specified, the format defaults to "kml." A list of formats supported by a particular map implementation that can be obtained by querying the map using the map.status channel (see map.status). Note that for this version of the Common Map Widget API, the only format that all map implementations *must* support is KML.
url:	URL from which to retrieve the feature data to load onto the map
zoom:	*true* if map should zoom to newly loaded feature data, *false* if not. Default is false.

Example:

```
{"featureId":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_link.kml
", "url":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_link.kml
", "overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data", "zoom": true}
```

Unplot Feature

Purpose:	Remove feature data from the map.
Channel:	`map.feature.unplot`
Payload:	`{overlayId: (optional), featureId: (required)}`
overlayId:	The id of the overlay in which the feature to be removed is found. If no overlayId is included, default overlay with id equal to sending widgets id is assumed.
featureId:	The id of the feature to be removed.

Example:

```
{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data", "featureId":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_link.kml
"}
```

Hide Feature

Purpose: Hide existing feature on the map.

Channel: `map.feature.hide`

Payload: `{overlayId: (optional), featureId: (required)}`

overlayId: The id of the overlay in which the feature to be hidden is found. If no overlayId is included, default overlay with id equal to sending widget's id is assumed.

featureId: The id of the feature to be hidden.

Example:
```
{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data", "featureId":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_link.kml
"}
```

Show Feature

Purpose: Have the map show previously hidden feature data.

Channel: `map.feature.show`

Payload: `{overlayId: (optional), featureID: (required), zoom: (optional)}`

overlayId: The id of the overlay in which the feature to be shown is found. If no overlayId is included, default overlay with id equal to sending widget’s id is assumed.

featureId: The id of the feature to be shown.

zoom: *true* if map should zoom to the shown feature, *false* if not. Default is false.

Example:
```
{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data", "featureId":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_link.kml
"}
```

Feature Selected

Purpose: “Select,” or report that object was selected.

Channel: `map.feature.selected`

Payload:
```
{selectedId: (optional*), selectedName: (optional*), featureId:
(required), overlayId: (optional)}
```

selectedId: The id of the actual clicked object (may be a sub-feature contained within the aggregate feature data with the given featureId).

selectedName: The name of the clicked object.

featureId: The id of the feature that contains the clicked object.

overlayId: The id of the overlay which contains the clicked object. If no overlayId is included, default overlay with id equal to sending widget's id is assumed.

Example:
```
{"selectedId": "12206", "selectedName": "12206", "featureId":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_content.
kml", "overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data"}
```

Notes: Although both selectedId and selectedName are optional, one must be passed in if a sub-feature is to be identified. Generally, selectedId is preferred, and selectedName is used when no selectedId is available.

map.view channels

Zoom

Purpose: Zoom the map to a particular range.

Channel: `map.view.zoom`

Payload: `{range: (required)}`

Range: The distance in meters from the map (note that most 2-D map renderers do not support infinite zoom, and the range will be translated into the nearest supported zoom level).

Example: `{"range": 100000}`

Center on Overlay

Purpose: Center the map on a particular overlay. The map may also be zoomed to show the entire overlay (if possible) or to show a given range.

Channel: `map.view.center.overlay`

Payload: `{overlayId: (optional), zoom: (optional)}`

overlayId: The id of the overlay to center on. If no overlayId is included, default overlay with id equal to sending widget's id is assumed.

zoom: If "auto," will adjust zoom to best-fit the overlay in the user's viewable area.
If a number, change zoom to specified range in meters.
If no zoom attribute is included, no zoom is performed.

Example: `{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-dd8c75434cf2.SAGE Public Access Data (no SAGE login required).Public SAGE data"}`

Center on Feature

Purpose: Center the map on a particular feature. The map may also be zoomed to show the entire feature (if possible) or to show a given range.

Channel: `map.view.center.feature`

Payload: `{overlayId: (optional), featureId: (required), zoom: (optional)}`

overlayId: The id of the overlay in which the feature to be centered on is found. If no overlayId is included, default overlay with id equal to sending widget's id is assumed.

featureId: The id of the feature to center on.

zoom: If "auto," map will adjust to best-fit the feature in the user's viewable area.
If a number, map will zoom to specified range in meters.
If no zoom attribute is included, no zoom is performed.

Example:

```
{"overlayId": "jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data", "featureId":
"https://sageearth.northcom.mil/arcgisserver/public/SAGE_link.kml
"}
```

Center on Location

Purpose: Center the map on a particular location. The map may also be zoomed as close as possible to the location or to a given range.

Channel: `map.view.center.location`

Payload: `{location: {lat: (required), lon: (required)} (required), zoom: (optional)}`

location: Location to be centered in map.

lat: The latitude value of the point.

lon: The longitude value of the point.

zoom: If "auto," map will adjust to zoom in to the given location as close as possible in the user's viewable area.
If a number, map will zoom to specified range in meters.
If no zoom attribute is included, no zoom is performed.

Example: `{"location": {"lat": 40.2205, "lon": -74.3579}, "zoom": 70250}`

Center on Bounds

Purpose: Center the map on a particular bounding box. The map may also be zoomed to show the entire bounds (if possible) or to show a given range.

Channel: `map.view.center.bounds`

Payload:
```
{bounds: {southWest: {lat: (required), lon: (required)}
(required), northEast: {lat: (required), lon: (required)}}
(required), zoom: (optional)}
```

bounds: Bounding box of area to be centered in map.

southWest:

lat: The latitude value of the southwest point.

lon: The longitude value of the southwest point.

northEast:

lat: The latitude value of the northeast point.

lon: The longitude value of the northeast point.

zoom: If “auto,” map will adjust to best fit the bounds in the user's viewable area.
If a number, map will zoom to specified range in meters.
If no zoom attribute is included, no zoom is performed.

Example:
```
{"bounds": {"southWest": {"lat": -16, "lon": -166}, "northEast":
{"lat": 75,"lon": -28}}, "zoom": "auto"}
```

Map Clicked

Purpose: “Click”, or report that map was clicked.

Channel: `map.view.clicked`

Payload: `{lat: (required), lon: (required)}`

lat: The latitude of the location that was clicked.

lon: The longitude of the location that was clicked.

Example: `{"lat":40.195659093364654,"lon":-74.28955078125}`

map.status channels

Request Map Status

Purpose: Request current status from the map. Map will send out requested map.status messages in response.

Channel: `map.status.request`

Payload: `{types: (optional)}`

types: An array of status types being requested. Currently only status types of “about,” “format,” and “view" are supported (future versions are expected to support a larger family of statuses, perhaps including “overlay,” “feature,” “selected”). If types attribute is not included, all status types will be generated.

Example: `{"types":["view", "about" ]}`

Map View Status

Purpose: Send out the current status of the map view.

Channel: `map.status.view`

Payload: `{requester: (optional), bounds: {southWest: {lat: (required), lon: (required)} (required), northEast: {lat: (required), lon: (required)}}, center: { lat: (required), lon: (required)} (required), range: (required)}.`

requester: Client that requested this status message be sent (if any). If no requester, message is being sent due to a map view change.

bounds: Bounding box of area visible on map.

southWest:

lat: The latitude value of the southwest point.

lon: The longitude value of the southwest point.

northEast:

lat: The latitude value of the northeast point.

lon: The longitude value of the northeast point.

center: The current center of the map.

lat: The latitude value of the point.

lon: The longitude value of the point.

range: The current distance, in meters, map is zoomed out.

Example:
```
{"requester": "217c086f-719f-928f-5e75-972530cf0db6", "bounds":
{"southWest": {"lat": 39.46164364205549, "lon": -
75.6134033203125}, "northEast": {"lat": 40.97575093157534,"lon":
-73.10302734375}},"center": {"lat": 40.2205,"lon": -74.3579},
"range": 137500}
```

Map Format Status

Purpose: Send out the list of data formats. This map implementation supports plotting feature data.

Channel: `map.status.format`

Payload: `{formats: [] (required)}`

format: An array of the formats that this map supports. Note that for this version of the Common Map Widget API, the only format that all map implementations *must* support is KML.

Example: `{"formats": ["kml"]}`

Map About Status

Purpose: Send out static information about this map implementation.

Channel: `map.status.about`

Payload:
```
{version: (required), mapType: (required), widgetName:
(required)}
```

version: The version number of the Common Map Widget API that this map widget supports.

type: The type of map widget (e.g. “2D” or “3D”).

widgetName: The name of the map widget.

Example:
```
{"version": "1.0.0", "type": “2D”, widgetName: "Common Map
Widget"}
```

Map Error Channels

Error

Purpose: Map Widget reports errors occurred when attempting to process any message.

Channel: `map.error`

Payload: `{sender: (required), type: (required), msg: (required), error: (required)}`

sender: Sender of message that caused error.

type: Type of message that caused error.

msg: The message that caused the error

error: A description of the error.

Example:

```
{"sender": "217c086f-719f-928f-5e75-972530cf0db6", type":
"map.feature.hide", "msg":
"{\"overlayId\":\"jc2cui.overlayWidget.69b08b71-7890-9096-13bf-
dd8c75434cf2.SAGE Public Access Data (no SAGE login
required).Public SAGE data\",\"featureId\":\"youCantFindMe\"}",
"error": "Unable to find feature youCantFindMe in overlay
jc2cui.overlayWidget.69b08b71-7890-9096-13bf-dd8c75434cf2.SAGE
Public Access Data (no SAGE login required).Public SAGE data"}
```

Other

Drag and Drop

Purpose: Drop an item on the map.

Channel: NA – See OWF drag drop API for details

Payload:

```
dragDropData: {overlayId: (optional), featureId: (required),
marker: {id: (optional), title: (optional), details: (optional),
iconUrl: (optional)} (optional), feature: {format: (required),
url: (optional), featureData: (optional)} (optional)}
```

overlayId: The id of the overlay the dropped item should be loaded into. If overlay with this id already exists, new item is merged into existing overlay; otherwise, new overlay will be created. If no overlayId is included, sending widget's id is used.

featureId: Unique identifier for the dropped feature. Note that feature ids must be unique within a given overlay. Reusing a feature id will be considered a reload, with the original feature data being removed and replaced by the new feature data.

marker: Information with which to create a new marker on the map.

- **id:** The id of this item.
- **title:** The title of the item.
- **details:** Detail text associated with the item that will appear in an info window.
- **iconUrl:** URL to an icon that represents this dropped item on the map. If no URL is included, a default icon is used.

feature: A URL or feature data to be plotted on the map.

- **format:** Data format (http header content type) of the given feature. A list of formats supported by a particular map implementation that can be obtained by querying the map using the map.status channel (see map.status).
- **url:** URL from which to retrieve the feature data to be loaded onto the map. If the URL does not resolve to a valid feature document, an error is generated.
- **featureData:** Feature data to be loaded into the map.

Example:

```
{dragDropLabel: 'Spot Report', dragDropData: {overlayId: 'Spot
Reports', marker: {title: 'A Spot Report!', details: 'Spot report
details here', iconUrl:
'http://localhost/mapWidget2/images/spotReport.gif'}}
```

Note: Although both marker and feature are optional, one must be present. Within a feature, at least one URL or featureData must be present.

If marker is included, the marker will be placed at the location of the drop. Data loaded via KML and URL will be placed at their natural location (equivalent to being loaded using map.feature.plot.* message)

Acronyms

API	Application Program Interface
GUI	Graphical User Interface
KML	Keyhole Markup Language
Lat	Latitude
Lon	Longitude
OWF	OZONE Widget Framework
URL	Uniform Resource Locator