Ryan Macklin

RyanMacklin@gmail.com

Developer documentation set sample

see resume for work history & qualifications

What's in this work sample packet?

This packet of sample SDK documentation covers four different types of developer pages:

- A learning example case walking developers through steps for handling a relatively complex process
- A static class for retrieving all or part of a general-purpose list of ID/name pairs
- A focused utility class for handling a small, frequently recurring sub-process with error and warning handling
- A specialized class that forms a complex API call and returns multiple varied objects if successful

Pages include commentary along the side, to show a little of my thought process.

About this fake product

Imagine a case where the State of California builds an animal disease logging system, designed to record, report on, and analyze various disease incidents within the state's agriculture.

Questions?

If you have questions after reading this, I'm happy to have a conversation about them.

Thank you!

Have a fantastic day!

RyanMacklin@gmail.com
see resume for work history & qualifications

Learning example case (page 1/2)

The first example doc in this packet is on how to generate an incident map—chosen because this shows an onboarding developer a few key classes and how our SDK implements server request needs.

This example is an unusual case, as it's very unlikely an onboarding developer will interact with the incident map subsystems. This makes for a good learning case because it touches on common topics used throughout the subsystems and shows how to think in the SDK's paradigm and mentally work around some technical debt leading to awkward design.

Focusing on new and onboarding developers: Every firm has turnover at some pooint, and new developers need to be onboarded as soon as possible. I wrote this example doc with a newly hired junior developer in mind as the baseline audience.

This audience mindset is also why the document ends on a passive set of further exercises. The point of that section isn't to create busy work. It's to stoke *curiosity* and *interest*—powerful tools for information retention.

This example continued on next page

Example case: generating an incident map

This example walks through generating an incident map for a particular region and condition: all bovine incidents in counties bordering Oregon, from the beginning of the current year through today.

You'll do the following, showcasing how to use various aspects of the SDK:

- Find the Animal ID for "bovine"
- Retrieve IDs for all counties bordering Oregon
- Set the search date range for the third quarter of 2017
- Make the call to retrieve the map
- Show the generated map and how many points of interest are found

Find the Animal ID for "bovine"

Normally we expect the user to supply the ID via UI interaction, but for this tutorial you'll hardcode the animal ID for bovine incidents. To find the ID for yourself, look it up in the Globals.Animals documentation. You should see the ID for "bovine" is 5.

```
var searchAnimalId = 5;
```

Retrieve IDs for all counties bordering Oregon

For this, we just call the GetBorderingOregon function in the static class Global.Counties.

```
var searchCountyIds = Global.Counties.GetBorderingOregon();
```

When called, GetBorderingOregon always returns the following set of three IDs: 7, 24, 46 (for Del Norte, Modoc, and Siskiyou counties respectively).

Set search date range

With a search specification of the third quarter of 2017 (June 1st to August 31st), that translates into searching from June 1st 2017 12:00am PDT until September 1st 2017 12:00am PDT. Create a new SearchDateTime object with those dates in the constructor.

```
var searchDates = new Utilities.SearchDateRange('June 1 2017',
'September 1 2017');
```

Per engineering specifications on dates, Pacific Time is always assumed when time zone is omitted, as is midnight when time is omitted.

RyanMacklin@gmail.com
see resume for work history & qualifications

This example continued from previous page

Learning example case (page 2/2)

"What about genuinely practical examples?" Any code that's of practical use—that is, being used in a product right now—might be the basis of an example article. But those are examples of code in action, meant to be copied an implemented.

We want exercises that walk through the SDK but are themselves suboptimal, as that gives mental room for developers to consider how to make improvements. That creates feelings of inspiration and investment early in a onboarding developer's tenure.

Make the call to retrieve the map

Next, create a new Map.IncidentMapGenerator object, populate it with your search parameters, and submit the call to retrieve map data.

```
var mapgen = new Map.IncidentMapGenerator();
mapgen.SetTypes.Animals(searchAnimalId);
mapgen.SetRegions.Counties(searchCountyIds);
mapgen.SetDateRange(searchDates);
var success = mapgen.Generate();
```

Show the map and incident count

Finally, render a small bit of HTML to show proof of a successful call, or of a failure message if the call wasn't successful.

```
if (success) {
     var url = mapgen.Map.Url;
     var count = mapgen.Markers.count();
     print("<div><img src=\"" + url + "\" alt=\"generated map\" /></
div>\n");
     print("<div>Marker count: " + count + "</div>\n");
} else {
     print("<div>IncidentMapGenerator call not successful</div>\n");
}
```

Note: In a real situation, you won't hand-code HTML. The visual components library handles rendering.

Full code

[Work sample note: to keep this example to 2 pages, not actually repeating the above code]

Beyond this exercise

Once you have this working, consider how to extend this code for these situations:

- Retrying after one server timeout
- Implement a simple UI to select parameters and handle invalid selections
- Display a second map by shifting the date range forward 3 months

End of this example article

RyanMacklin@gmail.com
see resume for work history & qualifications

Overview of a static class

This doc example covers a set of related functions and ID/name pair references drawn from a static data set.

Links to functions… when needed: This page is designed to make going to an individual function description largely unnecessary, as the target audience is expected to rarely need those details further spelled out.

But for times when details are needed, each function has a dedicated write-up including any notes. In an ideal world, our developer team would be able to add comments to each function, such as code references worth reviewing or answers to common questions. That's useful for community bonding as well as for my information architecture and maintenance efforts.

Globals.Counties

Static class for retrieving the system's global County name/ID pairs.

Functions

GetAdjacent (int CountyId)	Returns collection of IDs for counties adjacent to a given County
GetBorderingArizona ()	Returns collection of IDs for counties adjacent to Arizona
GetBorderingMexico ()	Returns collection of IDs for counties adjacent to Mexico
GetBorderingNevada ()	Returns collection of IDs for counties adjacent to Nevada
GetBorderingOregon ()	Returns collection of IDs for counties adjacent to Oregon
GetBorderingPacific ()	Returns collection of IDs for counties adjacent to the Pacific Ocean
GetList ()	Collection of all County name/ID pairs, sorted by name
GetName (int CountyId)	Returns county name
Search (string Name [, bool Partial])	Returns collection of all County name/ID pairs matching given string

County list

County ID	County Name
0	Alameda
1	Alpine
2	Amador
	[Work sample note: let's not bore you with a list of every county in California]
56	Yolo
57	Yuba

This list is automatically regenerated when the internal County table is updated.

End of this example article

RyanMacklin@gmail.com
see resume for work history & qualifications

Focused utility class (page 1/2)

This doc example details a class developed to handle a simple case: defining a search date range, with routine error and warning handing.

Intentionally missing for brevity purposes: To keep this section of the sample to two pages, I omitted synopses tables for attributes or functions. The next document shows synopses tables, as it's a longer example page.

Utilities.SearchDateRange

SearchDateRange handles logic for generating the date search portion of SQL queries, including default values and testing for invalid start/end date pairs.

Inherits from Utilities.DateRange.

Constructors

- SearchDateRange()
- SearchDateRange(datetime Start)
- SearchDateRange(datetime Start, datetime End)

Attributes

[Work sample note: attribute synopsis table would be here]

End [datetime, nullable]

The end of the datetime range. *Inherited from Utilities.DateRange.*

Set to null to use the default end time (per GetDefaultEnd). End may be set later than current datetime.

On change: If Start isn't null and End is set earlier than Start, a Warning.Date.EndRange warning is thrown. A warning is thrown rather than an error; errors occur only if this is still an issue when GetSqlPhrase is called.

Start [datetime, nullable]

The start of the datetime range. *Inherited from Utilities.DateRange.*

Set to null to use the default start time (per GetDefaultStart).

On change: If End isn't null and Start is set later than End, a Warning.Date.StartRange warning is thrown. That warning is also thrown if End is null and Start is set later than the default end datetime. A warning is thrown rather than an error; errors occur only if this is still an issue when GetSqlPhrase is called.

Functions

[Work sample note: function synopsis table would be here]

GetDefaultEnd ()

Inherited from Utilities.DateRange.

Returns the default End value: the current datetime.

This example continued on next page

RyanMacklin@gmail.com
see resume for work history & qualifications

This example continued from previous page

Focused utility class (page 2/2)

Referencing project specifications: For internal documentation like this, I advocate for linking to requirement and specification docs whenever an assumption, default, or other complex or unusual situation warrants a reference. In this example, there are separate docs for product, UX, and engineering requirements and specifications.

Having this easily accessibly gives developers a way to quickly look up the source of a given spec or decisions, either to better understand the context or to have an informed discussion on if that spec or decision is outdated.

GetDefaultStart ()

Overrides Utilities.DateRange.

Returns the default End value: 30 days prior to current End, at 12:00am.

GetSqlPhrase (string FieldName)

Inherited from Utilities.DateRange.

Returns a string of a parenthetical statement for use in SQL statements, as so:

```
([FieldName] >= '[StartDate]' AND [FieldName] < '[EndDate]')
```

Start and end dates are based on Start and End attributes, using default values for any null attribute.

This also serves as final error handling for invalid start/end date pairs. An Error.Date.InvalidStartEnd error is thrown if IsValid returns false.

If FieldName is empty or isn't alphanumeric, an Error.Date.InvalidFieldName error is thrown to prevent a malformed SQL query.

IsValid ()

Inherited from Utilities.DateRange.

Returns true if either:

- End and Start are both null
- End is null and Start is set earlier than or equal to the default end datetime
- Start is earlier then or equal to End

Returns false otherwise. Note that testing if Start or End is on or after the early bound is handled by the datetime object itself.

Additional notes

Default datetimes: The default map search start and end datetimes are set per product specifications.

Time zone: As defined in engineering specifications (and as implemented in Utilities.DateRange) time zones are in Pacific time—either PST or PDT depending on specified date. The system automatically detects standard or daylight savings time.

End of this example article

RyanMacklin@gmail.com
see resume for work history & qualifications

Object for a complex process (page 1/5)

This doc example covers an object that takes in highly variable parameters for a flexible API call. This example is long, to show how I approach such a situation within this framework and house style.

Since part of the point of this SDK is to enforce structure and system expectations, the objects themselves generate the JSON for API interactions. For complicated scenarios, that leads to objects like this one.

Managing cognitive glue and lengthy pages: I wouldn't blame anyone for skimming this section, given its length. As a developer, I would likely skim until I saw something relevant to my current objective.

This very notion is related to the concept of "cognitive glue." People are more likely to retain information they're trying to act upon, rather than general reading for the sake of broad understanding. Developer doc infrastructure needs to be designed with this in mind.

Map.IncidentMapGenerator

IncidentMapGenerator takes and validates a variety of inputs to generate a graphical map of an area with information on each incident found in that area over a given date range and other search filters.

Go to the product specifications for incident maps for detailed information.

Constructor

» IncidentMapGenerator()

Attributes

Markers [MapLegendMarker[]]	Collection of legend markers on map, each corresponding to one or more incidents (default: null)
Map [MapImageInfo]	Information on map retrieved, including file location or SVG code for the map itself (default: null)

Markers [MapLegendMarker[]]

Markers is null until Generate() is successfully executed. After that, it contains a collection of MapLegendMarker objects.

Coordinates.HeightPerc [float]	Vertical location for center of legend location, as a percentage of the image starting at the left
Coordinates.WidthPerc [float]	Horizontal location for center of legend location, as a percentage of the image starting at the top
Coordinates.X [int]	Horizontal pixel for center of legend location
Coordinates.Y [int]	Vertical pixel for center of legend location
Incidents [collection]	Collection of incidents linked to this legend marker, each containing a subset of a full Incident object:

AnimalType [int]	Primary animal type for incident
DiseaseType [in]	Primary disease type for incident
Geolocation [geo]	Longitude and latitude where incident took place
IncidentId [int]	ID of incident record
IncidentStatus [int]	Primary type of incident
LoggedDatetime [datetime]	Date and time incident was logged by an agent

This example continued on next page

RyanMacklin@gmail.com
see resume for work history & qualifications

This example continued from previous page

Object for a complex process (page 2/5)

Information architecture for less-than-optimal situations: I normally strive to avoid nested tables. They violate accessibility principles, as well as can be cumbersome to process for those who don't use screen accessibility software.

But for this case, I prioritized having this document be mostly useful document (albeit with accessibility issues) over holding the release until I could come to a perfect solution that works with Engineering's framework and needs.

ReportedDatetime [datetime]	Date and time incident was reported to happen
ShortDesc [string]	Up to first 100 characters of incident report

MapLegendMarker object [hide]

Map [MapImageInfo]

Map is null until Generate() is successfully executed. After that, it contains a MapImageInfo object.

MapImageInfo object [hide]

Size.Height [int]	Height of returned map image in pixels
Size.Width [int]	Width of returned map image in pixels
SvgCode [string]	SVG code. Only used if ImageType is Svg.
Type [Enum.Map.ImageType]	ImageType.File: provided image URL to reference ImageType.Svg: provided image SVG code to directly inject
Url [string]	URL of image. Only used if ImageType is File.

Functions

Generate ()	Submits search parameters and map specifications to server, retrieves map and legend data
SetDateRange (SearchDateRange range)	Sets date range search parameter
SetMapSpecs.Scale (int Scale)	Sets map scale to return
SetMapSpecs.Type (Enums.Map.SpecsType type)	Sets map type to return
SetRegions.Counties (varies)	Sets region search parameter to zero or more counties
SetRegions.Geo (varies)	Sets region search parameter to a geolocation square
SetRegions.Zips (varies)	Sets region search parameter to zero of more ZIP codes
SetTypes.Animals (varies)	Sets animal search parameter to zero or more animals types
SetTypes.Diseases (varies)	Sets disease search parameter to zero or more disease types
SetTypes.Incidents (varies)	Sets incident search parameter to zero or more disease types

This example continued on next page

RyanMacklin@gmail.com
see resume for work history & qualifications

This example continued from previous page

Object for a complex process (page 3/5)

Risks of overusing hyperlinks: In an older version if this document, much of text you see was hyperlinked. Every error linked to its prototype page. Every place where "specified threshold" exists linked to a project specification document. This has a good intention of directing readers to relevant information. But it has two significant drawbacks which causes me to be wary of overlinking.

First, it makes text on a page much harder to cognitively process. Constant formatting changes create distractions and increase the difficulty of reading a section linearly. It also adds difficulty for visually impaired readers, as screen readers point out each link throughout the reading experience.

Second, not all states of mind can handle a high volume of links. When readers find it difficult to concentrate or are in a state of overaction curiosity, they're unfortunately prone to following links that satisfy neurochemical desires for novelty and learning, which can detract from the core goal of developer documentation: enabling productivity.

Generate ()

Submits search parameters and map specifications to server, retrieving map and marker data. Returns true if able to make a connection and retrieve data.

If successful, Markers and Map attributes are populated with the resulting data. Otherwise, Markers and Map are unchanged.

Returns false if the server times out or an error is returned instead. This also throws the relevant error. Errors to anticipate:

- *Error.Data.TooLarge:* the API returns this error if the data volume exceeds specified threshold
- *Error.Geolocation.OutOfRange:* the geolocation is outside the specified geolocation range
- *Error.IncidentMap.CannotRetrieveImage:* the API is unable to retrieve or generate a map image
- *Error.IncidentMap.MutlipleRegions:* multiple types of Region parameters were attempted
- *Error.Server.ServerTimeout:* the server call exceeded the SDK's specified timeout threshold

SetDateRange (SearchDateRange range)

Sets date range search parameter.

Default value is an empty SearchDateRange object. If null is passed, date range is reset to the default value.

SetMapSpecs.Scale (int Scale)

Sets map scale to return. Value is number of pixels per mile.

Default value is 100, per incident map specifications. If null is passed, map scale is reset to the default value.

Error to anticipate:

- Error.Input.OutOfRange: scale is set to less than 1 or to over 1000

SetMapSpecs.Type (Enum.Map.SpecType type)

Sets map type to return, using Enum.Maps.SpecType

- SpecType.Terrain: provided image URL with markers on terrain
- SpecType.Vector: provided SVG code for markers and major features (area names, roads, borders, etc.)

Default value is SpecType.Terrain, per incident map specifications. If null is passed, map type is reset to the default value.

This example continued on next page

RyanMacklin@gmail.com
see resume for work history & qualifications

This example continued from previous page

Object for a complex process (page 4/5)

Using shortcodes for repeated text: In an ideal world, all of these repeated statements are document variables referenced via shortcode. This grants two significant benefits:

- The obvious: I only need to change one file or line to cover changes to all instances
- Slightly less obvious: I can search throughout the doc set for the shortcode, to find all cases where that text is used

For instance, I'd use a shortcode like {{Region.Constraint}} for the "Constraints: only..." paragraphs. Or for something that varies slightly such as the "Sets region search parameter..." lines, the shortcode could contain that variability as parameters: {{Region.DescLine "one or more counties"}} for the line in SetRegions.Counties, or {{Region.Desc "a geolocation square"}} for SetRegions.Geo.

I could nest this further, with all of the SetRegion functions generating the base text which uses shortcodes for each line. But there is a point to knowing when a refactoring technique is useful versus when it creates more work than benefit.

SetRegions.Counties (int CountyId)
SetRegions.Counties (int[] CountyIds)

Sets region search parameter to one or more counties.

Default value is an empty set. If null is passed, all prior entries are removed.

Constraint: only one type of Region can be set.

Errors and warnings to anticipate:

- *Error.Input.OutOfRange:* one or more IDs are outside of Global.Counties
- *Warning.IncidentMap.MutlipleRegions:* another Region parameter contains values
  While this is a warning, an error occurs if this issue isn't rectified before calling Generate()

SetRegions.Geo (float top, float left, float bottom, float right)

Sets region search parameter to a geolocation square.

Default value is an empty set. If null is passed, prior geolocation coordinates are removed.

Constraint: only one type of Region can be set.

Errors and warnings to anticipate:

- *Error.Geolocation.OutOfRange:* the geolocation is outside the specified geolocation range
- *Warning.IncidentMap.MutlipleRegions:* another Region parameter contains values
  While this is a warning, an error occurs if this issue isn't rectified before calling Generate()

SetRegions.Zips (int Zip)
SetRegions.Zips (int[] Zips)

Sets region search parameter to one or more ZIP codes.

Default value is an empty set. If null is passed, all prior entries are removed.

Constraint: only one type of Region can be set.

Errors and warnings to anticipate:

- *Error.Input.OutOfRange:* one or more ZIP codes are outside of Global.ZipCodes
- *Warning.IncidentMap.MutlipleRegions:* another Region parameter contains values
  While this is a warning, an error occurs if this issue isn't rectified before calling Generate()

This example continued on next page

RyanMacklin@gmail.com
see resume for work history & qualifications

This example continued from previous page

Object for a complex process (page 5/5)

No further commentary. If you've read up to this point, I'm very appreciative of your time!

SetTypes.Animals (int AnimalTypeId)
SetTypes.Animals (int[] AnimalTypeIds)

Sets animal search parameter to one or more animals type

Default value is an empty set. If null is passed, all prior entries are removed.

Clarification: Any number of search types can be passed. Passing multiple types narrows the results.

Error to anticipate:

» *Error.Input.OutOfRange:* one or more Animal IDs are outside of Global.Animals

SetTypes.Diseases (int DiseaseTypeId)
SetTypes.Diseases (int[] DiseaseTypeIds)

Sets disease search parameter to one or more disease types

Default value is an empty set. If null is passed, all prior entries are removed.

Clarification: Any number of search types can be passed. Passing multiple types narrows the results.

Error to anticipate:

» *Error.Input.OutOfRange:* one or more Disease IDs are outside of Global.Animals

SetTypes.Incidents (int IncidentsTypeId)
SetTypes.Incidents (int[] IncidentsTypeIds)

Sets incidents search parameter to one or more incident types

Default value is an empty set. If null is passed, all prior entries are removed.

Clarification: Any number of search types can be passed. Passing multiple types narrows the results.

Error to anticipate:

» *Error.Input.OutOfRange:* one or more Incident IDs are outside of Global.Incidents

Additional notes

For how the server processes provided inputs or for direct experimentation and testing, read the API documentation on MapUtils.GenerateIncidentMap.

End of this example article