Levrum DataBridge Post-Processing Scripts

Levrum DataBridge uses the Microsoft ClearScript engine to execute post-processing scripts written in JavaScript.

Levrum DataBridge connects to your Data Sources and combines them to create three different types of objects which are then combined to create JSON, CSV, and Microsoft Excel exports for use in your applications.

These objects are of type IncidentData, ResponseData, and TimingData, all of which are based on the AnnotatedData object. AnnotatedData objects store information in Dictionaries of type <string, object> named Data. These store the data from Field Mappings created with DataBridge. For example, in order to get the value of the Time Incident Data Mapping from the Incident object in a Per-Incident Script, you would use the JavaScript statement:

```
var Time = Incident.GetDataValue("Time");
```

These dictionaries can be accessed in JavaScript in two different ways. AnnotatedData objects contain three functions for interacting with their Data dictionarys, SetDataValue, GetDataValue, and RemoveDataValue.

You can also interact directly with the dictionaries using ClearScript Extended Host Functions.

For more information how to use Microsoft ClearScript please read their FAQtorial at https://microsoft.github.io/ClearScript/Tutorial/FAQtorial

DataBridge Output

DataSet, IncidentData, ResponseData, and TimingData all expose special properties for accessing specific entries within their Data dictionaries as follows:

DataSet

Id - String

IncidentData

Id - String
Time - DateTime
Location - String
Latitude - Double
Longitude - Double
Responses - DataSet

ResponseData

Parent - IncidentData
Id - String
TimingData - DataSet

TimingData

Parent - ResponseData
Name - String
Value - Double
Details - String
DateTime - DateTime
RawData - object

Post-Processing Host Objects

The following host objects are available in post-processing scripts:

Post-Loading Script

Incidents: DataSet of IncidentData objects created by the MapLoader
XHost: Microsoft ClearScript Extended Host Functions
Tools: Functions for quickly manipulating AnnotatedData entries
Debug: Functions for interacting with the DataBridge JS Debug Window
Logger: Functions for interacting with the DataBridge NLog logger
MapLoader: The DataBridge MapLoader class
ProgressInfo: Helper class for tracking processing progress

Per Incident Script

Incidents: DataSet of IncidentData objects created by the MapLoader
Incident: The current IncidentData object being processed
XHost: Microsoft ClearScript Extended Host Functions
Tools: Functions for quickly manipulating AnnotatedData entries
Debug: Functions for interacting with the DataBridge JS Debug Window
Logger: Functions for interacting with the DataBridge NLog logger
MapLoader: The DataBridge MapLoader class
ProgressInfo: Helper class for tracking processing progress

Final Processing Script

Incidents: DataSet of IncidentData objects created by the MapLoader
XHost: Microsoft ClearScript Extended Host Functions
Tools: Functions for quickly manipulating AnnotatedData entries
Debug: Functions for interacting with the DataBridge JS Debug Window
Logger: Functions for interacting with the DataBridge NLog logger
MapLoader: The DataBridge MapLoader class
ProgressInfo: Helper class for tracking processing progress

Post-Processing Host Types

The following host types are available in post-processing scripts:

IncidentData: IncidentData object
IncidentDataSet: DataSet
ResponseData: ResponseData object
ResponseDataSet: DataSet
TimingData: TimingData object
TimingDataSet: DataSet
MapLoaderErrorType: Enum containing error types.
MapLoaderError: Helper class for storing load errors
LogLevel: NLog LogLevel class
bool: Native C# boolean
double: Native C# double
string: Native C# string
DateTime: Native C# DateTime
TimeSpan: Native C# TimeSpan

```
// Possible error types NullIncidentId, NoResponseIdColumn, NullResponseId,
NullValue, BadValue, MergeConflict, LoaderException
```

```
var MapLoaderError = new MapLoaderError(MapLoaderErrorType.BadValue, "I found a bad value!");
```

Helper Object Functions

XHost

See Microsoft's reference at https://microsoft.github.io/ClearScript/Reference/html/T_Microsoft_ClearScript_ExtendedHostFunctions.htm

Tools

Functions used for manipulating AnnotatedData without making a large number of context switches

MergeDateTime(obj, key, date, time): Merge two DateTimes contained in AnnotatedData and store them in a single Entry

Example

```
// Usage MergeDateTime(AnnotatedData, outputKey, inputDate, inputTime)
Tools.MergeDateTime(Incident, "Time", "IncidentDate", "IncidentTime")
```

Debug

Functions used for outputting information to the DataBridge JavaScript debug window

Write(string): Write a string to the Debug Window
WriteLine(): Write a newline to the Debug Window
WriteLine(string): Write a string followed by a newline to the Debug Window
WriteObject(object): Convert an object to JSON and write it to the Debug Window

Example

```
Debug.Write("Moo");
Debug.WriteLine();
Debug.WriteLine("Moooooo");
var cow = {};
cow["says"] = "Mooooooooooooo";
Debug.WriteObject(cow);
```

Logger

See the NLog documentation at https://github.com/NLog/NLog/wiki/Tutorial for information on the Logger object

Example

```
Logger.Fatal("The most serious type of error");
Logger.Error("This is an error message");
Logger.Warning("WARNING BEEP BOOP");
Logger.Info("This is informative?");
Logger.Debug("No bugs allowed!");
```

MapLoader

Allows for direct access to the underlying MapLoader class

UpdateJSProgress(message, percentage): Used to send progress updates to the DataBridge's status bar
Incidents: DataSet containing the Incidents created by the DataBridge
IncidentsById: Dictionary<string, IncidentData> containing Incidents created by the DataBridge searchable by Id
ErrorRecords: List containing any errors encountered during the load process
Logger: Same as the Logger object
DebugHost: Same as the Debug object
CauseData: Data structure used for translating Codes to Categories and Types
Worker: BackgroundWorker running the MapLoader process
Map: The DataMap object used to generate IncidentData
bool Cancelling(): Returns true if the user has requested the load process be cancelled
LoadMap(DataMap): ***Main Function for Loading Data. Do Not Use***

```
MapLoader.UpdateJSProgress("Starting to do stuff", 0);
var error = new MapLoaderError(MapLoaderErrorType.BadValue, "I found a bad value!");
MapLoader.ErrorRecords.Add(MapLoaderError);
MapLoader.UpdateJSProgress("Finished doing stuff", 100);
```