Bridge : Data - Advanced: Data Types

!!! BEFORE WE START - THIS PDF IS OPTIONAL !!!

If you are an advanced user or want to start using PRO MODE then the following info will be very helpful. If you are new to the Bridge : Data extension then the information in this document can be skipped.

You will see in the next PDF that the Bridge : Data extension comes with a whole host of functions that is meant to hide the implementation details from you.

It is highly recommended that you study this PDF but it is entirely optional!

DATA RETURN TYPES

The format of the data that gets returned depends on which function you called to retrieve it.
The first entry is always going to be a header, you can count on that. Beyond that things get a little tricky…

When you are requesting multiple categories or multiple games you are going to have more than 1 node and must first find the node you want before you can start extracting any details from it.
Depending on which function you call the data structure of the results will differ greatly. For you to get to your data you first need to find it and for that you need to know how the data that is returned to you is formated.

In CML a node is denoted by enclosing it inside < and > and fields inside the node are enclosed within [and].
The CML returned to the ExtractMyDetails function from above would thus have looked something like this:

```
<DATA>
        <_CATEGORY_>
                [category] Personal
                [name] Neo
                [age] 30
                [position] (0,0,0)
                [rotation] (0,0,0,0)
```

Fetching a field and fetching a category results in the same structure and only differ in how many fields are inside the resulting node. In both cases entry 0 is the header "_DATA_" and array entry 1 is the _CATEGORY_ that contains your data.

<u>When fetching a single field:</u>

```
            <DATA>
                     [success] true
                     <_CATEGORY_>
                                [category] name
                                [requested field] value
```

<u>When fetching a single category:</u>

```
            <DATA>
                     [success] true
                     <_CATEGORY_>
                                [category] name
```

```
                        [field 1] value
                        [field n] value
                        ...
```

When fetching all the data of a game you will not only receive back one or more categories but these will be prefixed by another node called _GAME_ which means array entry 0 will be the header, 1 will be the _GAME_ node and your various categories will then follow starting from array position 2 (for the first game, at least. Other games's categories will obviously start at other array indexes).

When fetching a single game:

```
        <DATA>
        <_GAME_>
                [gid] value
                <_CATEGORY_>
                        [category] name
                        [field 1] value
                        [field n] value
                        ...
                <_CATEGORY_>
                        ...
```

When fetching everything then your data will contain one or more _GAME_ nodes under which it will store one or more _CATEGORY_nodes (or none at all).

When fetching everything:

```
        <DATA>
        <_GAME_>
                [gid] value
                <_CATEGORY_>
                        [category] name
                        [field 1] value
                        [field n] value
                        ...
                <_CATEGORY_>
                        ...
        <_GAME_>
                ...
```

CML offers you a plethora of functions to search for and filter content. As long as you are familiar with how the data is returned you should have no problems getting to your data.

PRO MODE

If you are familiar with CML then you can manipulate your data using all the freedom and flexibility that CML provides. You can filter your results using any of CML's functions but, to make it even easier for you and give you even more control over your data you can enable PRO MODE by setting WUData.WUDataPro to true anywhere in your project. Do this as early as possible so you know what format your data is returned in.

The Bridge : Data extension comes with a bunch of bloated functions to simplify life for people who are not familiar with CML. These functions have names that are easy to discern but requires the extra step of developers first storing the results from the Fetch* functions listed above and then calling one of these functions with the results of that operation as a parameter. When this second function returns a result you are ready to continue. In PRO MODE you just get the results from the Fetch* functions and do with it whatever you want to…

The reason for the distinction is because you can call your categories anything you like and thus there is no way to create a demo where users can copy paste code for their own use since everything will be hardcoded. Thus, when NOT in PRO MODE all categories always have the name "_CATEGORY_" and you can thus fetch a list of all _CATEGORY_ nodes to loop over BUT you must then loop over each to see what the value of the "category" field is in order to track down any specific category you want to work with.

In contrast, in PRO MODE the data is returned with the category as the node name (as CML was intended to work) so instead of you fetching all _CATEGORY_ nodes and then looping over each to see which one has a field called "category" with a value of "settings" (for example) and then saying "This is the category I was looking for" stead you can just say: var Settings = result.GetNodeOfType("settings"). This saves you all the looping and testing and just gives you exactly what you want using a single line of code.

Example normal data:

```
<DATA>
	<_CATEGORY_>
		[category] Personal
		[name] Neo
		[age] 30
		[position] (0,0,0)
		[rotation] (0,0,0,0)
```

Example PRO data:

```
<DATA>
	<Personal>
		[name] Neo
		[age] 30
		[position] (0,0,0)
		[rotation] (0,0,0,0)
	</Personal>
```

NOTE:

When PRO MODE is enabled the wrapper functions will no longer work. You will not need them, nor would you want to use them once you are adept at using CML but it is worth mentioning non the less.