Bridge : Data - Advanced: Wrapper Functions

PARSING FUN

The data returned from the server is in CML format meaning you can use any of CML's plethora of filtering and parsing functions to get to your data. For those of you who are not familiar with CML the Bridge : Data extension provides you with a new subset of search functions specifically designed for this asset to help you get started as easily as possible. This means that you now have two methods for getting to your data. If one doesn't suit your taste, simply use the other.

Example:

Assuming you called the server to fetch some data and you then stored the server's response in *raw_data*... Further assuming you have already extracted the _GAME_ node and stored it in *game*

<u>Using the Bridge : Data's custom functions:</u>
```
var categories =  WUData.ExtractCategoriesForGame(raw_data, game.ID);
```

<u>Using CML directly</u>
```
var categories = raw_data.Children(game.ID);
```

Both will yield the same result so the choice is up to you which one you prefer to use. Once you get familiar with the use of CML's functions you will find that you have far more control over the data that is returned to you. Use CML for total control over your returned data and use the wrapper functions to get started quickly.

AVAILABLE FUNCTIONS

To make use of the Extract* and Get* functions listed below you must have first contacted the server using one of the numerous Fetch* functions (ideally FetchEverything()), stored that result somewhere and must now pass *that* stored value as input to these functions.

NOTE:
The Get* and Extract* functions are wrapper functions created **solely** to help devs who are new to the asset to get up and running faster. **These functions DO NOT WORK IN PRO MODE !!**

List<cmlData> ExtractAllGames(CML raw_data)
Extracts only the _GAME_ nodes from raw_data. This provides you with the various game's gid values.
raw_data Valid CML data obtained from the server

cmlData ExtractFirstGame(CML raw_data)
Extracts only the first _GAME_ node from raw_data. This provides you with the game's gid value.
raw_data Valid CML data obtained from the server

ExtractAGameByField(CML raw_data, string field, string value)

Examines the global variables of all games found inside raw_data. If it finds one that contains the field in *field* and that field has the value specified by *value*, it returns it otherwise it returns null

raw_data
Valid CML data obtained from the server
field
The variable the global settings must contain. Eg. "name"
value
The value *field* must hold to be true. Eg. "PacMan"

cmlData ExtractAGameByName(CML raw_data, string name)

Examines the global variables of all games found inside raw_data. If it finds one that contains a field called "name" and it's value matches the value in *name* it returns that game node, otherwise it returns null

raw_data
Valid CML data obtained from the server
name
The value this game's global variable "name" must hold to be true. Eg. "PacMan"

List<cmlData> ExtractCategoriesForGame(CML raw_data, int ID)
List<cmlData> ExtractCategoriesByGameName(CML raw_data, string name)

Extract the categories for a specific game so you can access the fields contained within them

raw_data
Valid CML data obtained from the server
ID
The ID of the _Game_ node you want the categories of. Note: Not the gid variable, but the node's actual ID!
Example:
cmlData PacMan = WUData.ExtractAGameByName(raw_data, PacMan");
if (null != PacMan) categories = ExtractCategoriesForGame(raw_data, PacMan.ID);
name
First calls ExtractAGameByName then returns the categories found for that game node

CMLData ExtractCategoryFromGame(CML raw_data, string category_name = "", int id = 1)
CMLData ExtractCategoryFromGameName(CML raw_data, string category_name = "" , string game_name = "Global")

Extracts only the specified category from the game

raw_data
Valid CML data obtained from the server
name
The name of the category you want to extract
id
The ID of the game node. Note: Not the gid value, but the game node's ID.
game_name
The game to find via ExtractAGameByName before the category is extracted from it

```
string GetField(List<CMLData> categories, string field_name, string cat = "")
int GetFieldi(List<CMLData> categories, string field_name, string cat = "")
float GetFieldf(List<CMLData> categories, string field_name, string cat = "")
```

Extracts a specific field from an extracted list of categories and returns it formatted as string or numeric data types

categories
A list of categories extracted from a game
field_name
The field to look for
cat
The category to look in for the field. Defaults to global category

```
string GetField(CML data, string field_name, string cat = "", int gid = 0)
float GetFieldf(CML data, string field_name, string cat = "", int gid = 0)
int GetFieldi(CML data, string field_name, string cat = "", int gid = 0)
string GetField(CML data, string field_name, string cat = "", string game_name = "")
float GetFieldf(CML data, string field_name, string cat = "", string game_name = "")
int GetFieldi(CML data, string field_name, string cat = "", string game_name = "")
```

Extracts a specific field from the server's response and returns it formatted as string or numeric data types

data
Valid CML data obtained from the server
field_name
The field to look for
cat
The category to look in for the field. Defaults to global category
gid
The game this variable belongs to. Defaults to global settings
game_name
The name of the game this variable belongs to. Defaults to global settings