```
/*
Performance Data Import/Entry Tool:
Takes csv data from performance tests and inserts into monthly report sheets.
Configure settings in Settings.xml with Drive Notepad

Settings.xml looks like this:
<settings>
	<mapsToAnalyze>
		<map>be1</map>
		<map>be2</map>
		<map>be3</map>
		<map>qu1</map>
		<map>qu2</map>
	</mapsToAnalyze>
	<folders>
		<folder>2017_8.29</folder>
	</folders>
	<filenames>
		<file>2017.8.29_report</file>
	</filenames>
	<machines>
	    <machine name="Laptop">laptop</machine>
	    <machine name="Tester XP">testerxp</machine>
	    <machine name="Tester 7">hztester7-pc</machine>
	    <machine name="Tester 5">hztester5-pc</machine>
	    <machine name="Tester 3A">tester3a</machine>
	    <machine name="Tester 3B">tester3b</machine>
	    <machine name="Tester 6">tester6-pc</machine>
	    <machine name="Tester 4">tester4-pc</machine>
	    <machine name="Tester">tester</machine>
	    <machine name="Tester 25">tester25-pc</machine>
	</machines>
	<game>Paladins</game>
</settings>

Use local (PowerShell-Based) PerfTestDataCopy.ps1 and follow prompts to get
data in the right place

by: Matt Canei, Tech Artist, 2017
*/
var mapsToAnalyze = [];
var gameName;
var folderNames = [];
var fileNames = [];
var machines = [];
```

```
var friendlyNames = [];
var dictComputers = {};
var spreadsheetFile;

var dataFolder;
var arrayPath = [];
var memorySumHeading = "TEST";
var defaultFolder = "Min.Spec.Machine";
var dataFolderName = "VMMap_Data";

function Main() {
    //populate XML vars before finding Folders, redefine arrayPath since vars
are from XML
    ImportXMLSettings();
    arrayPath = [defaultFolder, gameName, folderNames[0]]

    var currFolder = findFolder(arrayPath[2]);
    if (currFolder == -1) {
        Logger.log("Epic Massive Fail: Defcon 5");
    }
    Logger.log("Folder : " + currFolder);

    getDataFolder(currFolder);

    //get spreadsheet file to prepare for CSV loop
    spreadsheetFile = getFiles(currFolder, fileNames[0]);
    Logger.log("Spreadsheet File: " + spreadsheetFile);
    ProcessCSVs();
}

// path is 3rd level folder, assumes global variable arrayPath exists with
the rest of path in [0,1]
function findFolder(path) {
    var folders, parent, gParent, folderArray = [];

    //reassemble path array (array as Arg not supported)
    folderArray = arrayPath;
    folders = DriveApp.getFoldersByName(path);
    while (folders.hasNext()) {
        var folder = folders.next();
        Logger.log(folderArray);
        Logger.log('GetName ' + folder);

        var stepperFolder = folder.getParents();
        var foundit = false;
```

```
        //set up parent checks
        for (var i = folderArray.length - 2; i > 0; i--) {
            foundit = false;
            // checking this parent matches
            while (stepperFolder.hasNext()) {
                currParent = stepperFolder.next();
                if (currParent == folderArray[i]) {
                    foundit = true;
                    stepperFolder = currParent.getParents();
                    //return folder;
                    break;
                }
            }
            if (foundit == false) {
                break;
            }
        }

        if (foundit == false) {
            continue;
        }
        return folder;
    }

    return -1;

}

function getDataFolder(dFolder) {
    var dFolders = dFolder.getFolders();
    while (dFolders.hasNext()) {
        //if (dFolders.next() == dataFolderName) {
        dataFolder = dFolders.next();
        Logger.log('Data Folder Set: ' + dataFolder);
        // }
    }
}

function getFiles(cFolder, cFile) {
    var files = cFolder.getFilesByName(cFile);
    while (files.hasNext()) {
        var file = files.next();
        Logger.log(file);
    }
```

```
    return file;
}

//designed to loop through machine folders, loop through CSVs and insert into
spreadsheet.
function ProcessCSVs() {
    var csvPath = [];
    var currCSVFolder;
    for (var i = 0; i < machines.length; i++) {
        //loop through machine-named folders and store each in mFolder
        var mFolders = dataFolder.getFoldersByName(dictComputers[i].machine);

        while (mFolders.hasNext()) {
            var mFolder = mFolders.next();
            Logger.log('mFolder = ' + mFolder);

            var mChildren = mFolder.getFolders();
            var mChild = mChildren.next();
            Logger.log('mChild = ' + mChild);

            for (var m = 0; m < mapsToAnalyze.length; m++) {
                //loop through csv files and get each file from this month's
capture
                var files = mChild.getFilesByName(mapsToAnalyze[m] + '.csv');
                while (files.hasNext()) {
                    var csvFile = files.next();
                    Logger.log(csvFile);

                    //files obtained now do the magics
                    var csvData;
                    //var csvFile.open
                    var csv = csvFile.getBlob().getDataAsString();

                    csvData = Utilities.parseCsv(csv);
                    Logger.log(csvData[8][1]);

                    //take csv info and insert into spreadsheet in
appropriate place
                    var cSheet = SpreadsheetApp.open(spreadsheetFile);
                    Logger.log('csheet = ' + cSheet);


                    //take current data from processing loops and send to
dataEntry (ex. takes spreadsheet ref, memory data, current capture name and
machine name
```

```
                    dataEntry(cSheet, csvData[8][1], mapsToAnalyze[m],
dictComputers[i].friendlyName);

                }


            }

        }

    }
}

function dataEntry(spreadsheet, csvData, currentMap, currentMachine) {
    Logger.log(spreadsheet + '  ' + csvData);
    var dataSheet = spreadsheet.getSheetByName("data");
    var sheet = spreadsheet.setActiveSheet(dataSheet);

    //loop throw columns and rows to search for currentMap (capture Name
"mi1") and current Machine's friendly name (ex. Tester 25)
    for (var i = 1; i < 18; i++) {
        for (var j = 1; j < 18; j++) {
            var colValue = sheet.getRange(2, i).getValue();
            var rowValue = sheet.getRange(j, 1).getValue();

            if (colValue == currentMap && rowValue == currentMachine) {
                sheet.setActiveRange(sheet.getRange(j, i)).setValue(csvData);
            }
        }
    }
    Logger.log("Current Capture: " + currentMap + " " + currentMachine);

}


function ImportXMLSettings() {
    // get file PerfDataSettings.xml
    var fileName = 'Settings.xml';
    var fileID = '0By6BK3E1q7XQbVFGaFdfbnJ2dFk';
    //var file = DriveApp.getFilesByName(fileName);

    var file = DriveApp.getFileById(fileID);

    //open & read content of xml
    var contentXML = file.getAs(MimeType.HTML).getDataAsString();
```

```
    var document = XmlService.parse(contentXML);
    var root = document.getRootElement();
    // get maps to Analyze section - subset of maps that reports will be
generated on
    var projects = root.getChild('mapsToAnalyze').getChildren();
    var elements;
    var sheet;
    var computers;

    elements = root.getChild('mapsToAnalyze').getChildren('map');
    mapsToAnalyze = [];
    for (var j = 0; j < elements.length; j++) {
        mapsToAnalyze.push(elements[j].getText());
    }
    // get memory stats - the stat names which will be summed up
    elements = root.getChild('folders').getChildren('folder');
    folderNames = [];
    for (var j = 0; j < elements.length; j++) {
        folderNames.push(elements[j].getText());
    }

    gameName = root.getChild('game').getText();

    // get memory stats - the stat names which will be summed up
    sheet = root.getChild('filenames').getChildren('file');
    fileNames = [];
    for (var j = 0; j < sheet.length; j++) {
        fileNames.push(sheet[j].getText());
    }

    computers = root.getChild('machines').getChildren('machine');
    for (var i = 0; i < computers.length; i++) {
        var name = computers[i]
        var friendlyName = name.getAttribute('name').getValue();
        friendlyNames.push(friendlyName);
    }
    machines = [];
    for (var j = 0; j < computers.length; j++) {
        machines.push(computers[j].getText());
    }

    for (var i = 0; i < machines.length; i++) {
        //dictComputers.push({friendlyName: friendlyNames[i], machine:
machines[i]});
        dictComputers[i] = {
```

```
            friendlyName: friendlyNames[i],
            machine: machines[i]
        };
        //return dictComputers;
    }


    //set folder array path after XML data is grabbed
    var arrayPath = ['Min.Spec.Machine', gameName, folderNames[0]]

}
```