

AIQ Integration

startStep(String: step name)

Starts an AIQ step for summary reporting. Starting a new step will close the previous one automatically.

endFailingStep(String: failure)

End an AIQ step and mark it as failed with failure reason.

startSummaryStep(String: step name)

Starts an AIQ summary step for summary reporting. Must make pair with the corresponding endSummaryStep/endFailingSummaryStep.

Screen%20Shot%202021-04-28%20at%2015.24.39%20(1).png

endSummaryStep(String: step name)

Ends AIQ summary step for summary reporting. Must make pair with the corresponding startSummaryStep.

endFailingSummaryStep(String: failure)

Starts an AIQ step for summary reporting. Must make pair with the corresponding startSummaryStep.

HashDPL(String: filename)

Creates variables starting with \$ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable \$page_index. It will use the data in the first line after the header line of the DPL.

HashDPL(String: filename, Integer index)

Creates variables starting with \$ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable \$page_index. It will use the

data in the line number defined by the index.

EncryptedDPL(String: filename)

Creates variables starting with \$ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable \$page_index. It will use the data in the first line after the header line of an Encrypted .CSV file.

EncryptedDPL(String: filename, Integer index)

Creates variables starting with \$ based on the file name, the values with spaces are replaced with an underscore. e.g. the value "page index" will create variable \$page_index. It will use the data in the line number defined by the index of an Encrypted .CSV file.

```
if (typeof $accesskey == "undefined") {  
    EncryptedHashDPL("EncryptedAWSredentials.csv", "AppvncPassw!*6%");  
}
```

locker.get(Object: key) → Object

Calls AIQ TheLocker and returns the value stored for the specified key.

locker.put(Object: key ,Object:value)

Calls AIQ TheLocker and stores the value for the specified key.

addToLocalWhiteboard(Object: key, Object:value)

Calls AIQ Whiteboard DPL and stores the value for the specified key.

```
addToLocalWhiteboard("x", 1);  
var x = getFromLocalWhiteboard("x");  
log(x);
```

getFromLocalWhiteboard(Object: key) → Object:value

Calls AIQ Whiteboard DPL and gets the value stored for the specified key.

```
addToLocalWhiteboard("x", 1);  
var x = getFromLocalWhiteboard("x");  
log(x);
```

clearLocalWhiteboard()

Calls AIQ Whiteboard DPL and clears all the key-value pairs stored.

```
clearLocalWhiteboard();  
addToLocalWhiteboard("x", 1);  
var x = getFromLocalWhiteboard("x");  
log(x);
```

Revision #1

Created 2026-07-24 14:33:29 UTC by Admin

Updated 2026-07-24 14:33:29 UTC by Admin