

system.dnp3.directOperateAnalog

This function is used in **Python Scripting**.

Description

Issues a Select-And-Operate command to set an analog value in an analog output point.

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.directOperateAnalog(deviceName, index, value, [variation])

- Parameters

- String** deviceName - The name of the DNP3 device driver.
- Integer** index - The index of the object to be modified in the outstation.
- Numeric** value - The analog value that is requested (of type integer, short, float, or double).
- Integer** variation - The DNP3 object variation to use in the request. [optional]

- Returns

- Integer** - The DNP3 status code of the response.

- Scope

- Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example sets the analog output at index 0 to the
# double value 3.14.

system.dnp3.directOperateAnalog("Dnp3", 0, 3.14)
```

Code Snippet

```
# This example sets the analog output at index 2 to the
# integer value 300.

system.dnp3.directOperateAnalog("Dnp3", 2, 300)
```

Code Snippet

```
# This example sets the analog output at index 15 to the
# short value 33. The value sent in the request is converted
# for the object variation, 2.

system.dnp3.directOperateAnalog("Dnp3", 15, 33.3333, variation=2)
```

Code Snippet

```
# This example sets the analog output at index 1 to the
# float value 15.0.  The value sent in the request is converted
# for the object variation, 3.

system.dnp3.directOperateAnalog("Dnp3", index=1, value=15, variation=3)
```

Keywords

```
system dnp3 directOperateAnalog, dnp3.directOperateAnalog
```

system.dnp3.directOperateBinary

This function is used in **Python Scripting**.

Description

Issues a Direct-Operate command for digital control operations at binary output points (CROB).

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.directOperateBinary(deviceName, indexes, opType, tcCode, count, [onTime], [offTime])

- Parameters

String deviceName - The name of the DNP3 device driver.

List indexes - A list of indexes of the objects to be modified in the outstation.

Integer opType - The type of the operation: 0=NUL, 1=PULSE_ON, 2=PULSE_OFF, 3=LATCH_ON, 4=LATCH_OFF.

Integer tcCode - The Trip-Close code, used in conjunction with the opType: 0=NUL, 1=CLOSE, 2=TRIP.

Integer count - The number of times the outstation shall execute the operation.

Long onTime - The duration that the output drive remains active, in millis. [optional]

Long offTime - The duration that the output drive remains non-active, in millis. [optional]

- Returns

The **DNP3 status code** of the response, as an integer.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example shows latching off 3 binary output points with the Direct-Operate command.  
system.dnp3.directOperateBinary("Dnp3", [0, 1, 2], 4)
```

Code Snippet

```
# This example sets a binary output point at index 3 to pulse at 5 second intervals  
# with the Direct-Operate command.  
  
system.dnp3.directOperateBinary("Dnp3", [3], 2, 2, onTime=5000, offTime=5000)
```

Keywords

system dnp3 directOperateAnalog, dnp3.directOperateAnalog

system.dnp3.freezeAnalogs

This function is used in **Python Scripting**.

Description

Issues a freeze command on the given analog outputs.

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.freezeAnalogs(deviceName, [indexes])

- Parameters

String deviceName - The name of the DNP3 device driver.

List indexes - A list of specific indexes on which to issue the freeze command. An empty list can be passed to freeze all analogs.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example shows a request to freeze all analog inputs in the outstation.  
system.dnp3.freezeAnalogs("Dnp3", [])
```

Code Snippet

```
# This example shows a request to freeze analog inputs at indexes 1, 3, and 5.  
system.dnp3.freezeAnalogs("Dnp3", [1, 3, 5])
```

Keywords

system dnp3 freezeAnalogs, dnp3.freezeAnalogs

system.dnp3.freezeAnalogsAtTime

This function is used in **Python Scripting**.

Description

Issues a freeze command on the given analog outputs at the given time for the specified duration.

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.freezeAnalogsAtTime(deviceName, absoluteTime, intervalTime, [indexes])

- Parameters

String deviceName - The name of the DNP3 device driver.

Integer absoluteTime - The absolute time at which to freeze, in millis.

Integer intervalTime - The interval at which to periodically freeze, in millis.

List indexes - A list of specific indexes on which to issue the freeze command. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This example shows a request to freeze analog inputs at indexes 2 and 4,  
# 5 minutes from the current time, with no interval.  
from time import *  
  
fiveMikes = (60 * 1000 * 5) + int(time()) * 1000 #ms  
system.dnp3.freezeAnalogsAtTime("Dnp3", fiveMikes, 0, [2, 4])
```

Keywords

system dnp3 freezeAnalogsAtTime, dnp3.freezeAnalogsAtTime

system.dnp3.freezeCounters

This function is used in **Python Scripting**.

Description

Issues a freeze command on the given counters.

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.freezeCounters(deviceName, [indexes])

- Parameters

String deviceName - The name of the DNP3 device driver.

List indexes - A list of specific indexes on which to issue the freeze command. An empty list can be passed to freeze all counters. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example shows a request to freeze all counters in the outstation.  
system.dnp3.freezeCounters("Dnp3", [])
```

Code Snippet

```
# This example shows a request to freeze counters at indexes 1, 3, and 5.  
system.dnp3.freezeCounters("Dnp3", [1, 3, 5])
```

Keywords

system dnp3 freezeCounters, dnp3.freezeCounters

system.dnp3.freezeCountersAtTime

This function is used in **Python Scripting**.

Description

Issues a freeze command on the given counters at the given time for the specified duration.

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.freezeCountersAtTime(deviceName, absoluteTime, intervalTime, [indexes])

- Parameters

String deviceName - The name of the DNP3 device driver.

Integer absoluteTime - The absolute time at which to freeze, in millis.

Integer intervalTime - The interval at which to periodically freeze, in millis.

List indexes - A list of specific indexes on which to issue the freeze command. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This example shows a request to freeze counters at indexes 2 and 4,
# 5 minutes from the current time, with no interval.
from time import *

fiveMikes = (60 * 1000 * 5) + int(time()) * 1000 #ms
system.dnp3.freezeCountersAtTime("Dnp3", fiveMikes, 0, [2, 4])
```

Keywords

system dnp3 freezeCountersAtTime, dnp3.freezeCountersAtTime

system.dnp3.selectOperateAnalog

This function is used in **Python Scripting**.

Description

Issues a Select-And-Operate command to set an analog value in an analog output point.

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.selectOperateAnalog(deviceName, index, value, [variation])

- Parameters

String deviceName - The name of the DNP3 device driver.

Integer index - The index of the object to be modified in the outstation.

Number value - The analog value that is requested (of type integer, short, float, or double).

Integer variation - The DNP3 object variation to use in the request. [optional]

- Returns

The [DNP3 status code](#) of the response, as an integer.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example sets the analog output at index 0 to the double value 3.14.  
  
system.dnp3.selectOperateAnalog("Dnp3", 0, 3.14)
```

Code Snippet

```
# This example shows setting the analog output at index 2 to the  
# integer value 300.  
  
system.dnp3.selectOperateAnalog("Dnp3", 2, 300)
```

Code Snippet

```
# This example shows setting the analog output at index 15 to the  
# short value 33. The value sent in the request is converted  
# for the object variation, 2.  
  
system.dnp3.selectOperateAnalog("Dnp3", 15, 33.3333, variation=2)
```

Code Snippet

```
# This example shows setting the analog output at index 1 to the
# float value 15.0. The value sent in the request is converted
# for the object variation, 3.

system.dnp3.selectOperateAnalog("Dnp3", index=1, value=15, variation=3)
```

Keywords

```
system dnp3 selectOperateAnalog, dnp3.selectOperateAnalog
```

system.dnp3.selectOperateBinary

This function is used in **Python Scripting**.

Description

Issues a Select-And-Operate command for digital control operations at binary output points (CROB).

Client Permission Restrictions

Permission Type: DNP3 Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.dnp3.selectOperateBinary(deviceName, indexes, opType, tcCode, [count], [onTime], [offTime])

- Parameters

String deviceName - The name of the [DNP3 device driver](#).

List indexes - A list of indexes of the objects to be modified in the outstation.

Integer opType - The type of operation: 0=NUL, 1=PULSE_ON, 2=PULSE_OFF, 3=LATCH_ON, 4=LATCH_OFF.

Integer tcCode - The Trip-Close code, used in conjunction with the opType: 0=NUL, 1=CLOSE, 2=TRIP.

Integer count - The number of times the outstation shall execute the operation. [optional]

Integer onTime - The duration that the output drive remains active, in millis. [optional]

Integer offTime - The duration that the output drive remains non-active, in millis. [optional]

- Returns

Integer - The [DNP3 status code](#) of the response.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example shows latching on 3 binary output points with the Select-And-Operate command.  
  
system.dnp3.selectOperateBinary("Dnp3", [0, 1, 2], 3)
```

Code Snippet

```
# This example shows setting a binary output point at index 3 to pulse at 5 second intervals  
# with the Select-And-Operate command.  
  
system.dnp3.selectOperateBinary("Dnp3", [3], 1, 2, count=2, onTime=5000, offTime=5000)
```

Keywords

system dnp3 selectOperateBinary, dnp3.selectOperateBinary

system.eam

EAM Functions

The following functions give you access to view EAM information from the Gateway.

[In This Section ...](#)

system.eam.getGroups

This function is used in [Python Scripting](#).

Description

Returns the names of the defined agent organizational groups in the Gateway.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.eam.getGroups()

- Parameters

Nothing

- Returns

A list of group names.

- Scope

Gateway, Vision Client, Perspective Session

Examples

Code Snippet

```
# Return and print all of the EAM groups.  
groups = system.eam.getGroups()  
for group in groups:  
    print group
```

Keywords

system eam getGroups, eam.getGroups

system.eam.queryAgentHistory

This function is used in **Python Scripting**.

Description

Returns a list of the most recent agent events

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.eam.queryAgentHistory(groupIds, agentIds, startDate, endDate, limit)

- Parameters

[List](#) groupIds - A list of groups to restrict the results to. If not specified, all groups will be included.

[List](#) agentIds - A list of agent names to restrict the results to. If not specified, all agents will be allowed.

[Date](#) startDate - The starting time for history events. If null, defaults to 8 hours previous to now.

[Date](#) endDate - The ending time for the query range. If null, defaults to "now".

[int](#) limit - The limit of results to return. Defaults to 100. A value of 0 means "no limit".

- Returns

[Dataset](#) - A dataset with columns id, agent_name, agent_role, event_time, event_category, event_type, event_source, event_level, event_level_int, and message, where each row is a new agent event.

- Scope

Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Querying for Agent Task History

```
# This script loops through each row of the dataset and grab out every value from that row and assign it to a matching variable. Those variables can then be used in some way.
results=system.eam.queryAgentHistory()
for row in range(results.rowCount):
    eventId=results.getValueAt(row, "id")
    agentName=results.getValueAt(row, "agent_name")
    agentRole=results.getValueAt(row, "agent_role")
    eventTime=results.getValueAt(row, "event_time")
    eventCategory=results.getValueAt(row, "event_category")
    eventType=results.getValueAt(row, "event_type")
    eventSource=results.getValueAt(row, "event_source")
    eventLevel=results.getValueAt(row, "event_level")
    eventLevelInt=results.getValueAt(row, "event_level_int")
    message=results.getValueAt(row, "message")
    #Can include some code here to use the variables in some way for each row.
```

Code Snippet - Querying for Agent Task History

```
# This script grabs the agent event history from agents called Agent1, Agent2, Agent3, and will then place the data into a table on the same window.
```

```
results=system.eam.queryAgentHistory(agentIds=[ "Agent1", "Agent2", "Agent3" ])
event.source.parent.getComponent('Table').data = results
```

Keywords

system eam queryAgentHistory, eam.queryAgentHistory

system.eam.queryAgentStatus

This function is used in **Python Scripting**.

Description

Returns the current state of the matching agents.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.eam.queryAgentStatus(groupIds, agentIds, isConnected)

- Parameters

[List](#) groupIds - A list of groups to restrict the results to. If not specified, all groups will be included.

[List](#) agentIds - A list of agent names to restrict the results to. If not specified, all agents will be allowed.

[Boolean](#) isConnected - If True, only returns agents that are currently connected. If False, only agents that are considered down will be returned, and if not specified, all agents will be returned.

- Returns

[Dataset](#) - A dataset with columns AgentName, NodeRole, AgentGroup, LastCommunication, IsConnected, IsRunning, RunningState, RunningStateInt, LicenseKey, and Version, where each row is a new agent.

Possible values for RunningState and RunningStateInt are: 0 = Disconnected, 1 = Running, 2 = Warned, 3 = Errorred

- Scope

Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Querying Agent Status Information

```
# This script loops through each row of the dataset and grabs out every value from that row and assigns it to a matching variable. Those variables can then be used in some way.
results=system.eam.queryAgentStatus()
for row in range(results.rowCount):
    agentName=results.getValueAt(row, "AgentName")
    nodeRole=results.getValueAt(row, "NodeRole")
    agentGroup=results.getValueAt(row, "AgentGroup")
    lastComm=results.getValueAt(row, "LastCommunication")
    isConnected=results.getValueAt(row, "IsConnected")
    isRunning=results.getValueAt(row, "IsRunning")
    runningState=results.getValueAt(row, "RunningState")
    runningStateInt=results.getValueAt(row, "RunningStateInt")
    licenseKey=results.getValueAt(row, "LicenseKey")
    platformVersion=results.getValueAt(row, "Version")
    # Can include some code here to use the variables in some way for each row like printing each variable to the console.
```

Code Snippet - Querying Agent Status Information

```
# This script grabs status information from agents called Agent1, Agent2, Agent3, and then places the data into a table on the same window.
results=system.eam.queryAgentStatus(agentIds=[ "Agent1", "Agent2", "Agent3"])
event.source.parent.getComponent('Table').data = results
```

Keywords

system eam queryAgentStatus, eam.queryAgentStatus

system.eam.runTask

This function is used in **Python Scripting**.

Description

Takes the name of a task as an argument as a string (must be configured on the Controller before hand), attempts to execute the task.

To run in the client, the user needs a [role-based permission](#). This permission is disabled by default.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.eam.runTask(taskname)

- Parameters

String taskname - Name of the task to run. If more than one task has this name, an error will be returned.

- Returns

A [UIResponse](#) with a list of infos, errors, and warnings. The [UIResponse](#) object is functionally a list of runTask objects.

- Scope

Gateway, Vision Client, Perspective Session

UI Response

The "UIResponse" is an object containing three lists, each containing different logging information about the task that was run. The contents of the lists are accessible from the getter methods.

- [getWarns\(\)](#) - Returns a list of warning messages that were encountered during the task
- [getErrors\(\)](#) - Returns a list of error messages that were encountered during the task
- [getInfos\(\)](#) - Returns a list of "info" messages that were encountered during the tasks.

These messages represent normal logging events that occurred during the task, and can be useful when to visualize the events that lead up to a task failure.

Examples

Code Snippet

```
# Execute a task called 'Collect Backup'.
taskName = "Collect Backup"
response = system.eam.runTask(taskName)
```

Code Snippet

```
# Execute a task and display the responses from it.

# Create a function to print out the responses in a nice format.
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
```

```
print " None"

# Run the task.
taskName = "Collect Backup"
response = system.eam.runTask(taskName)

# Print out the returned Warnings (if any).
warnings = response.getWarns()
print "Warnings are:"
printResponse(warnings)

# Print out the returned Errors (if any).
errors = response.getErrors()
print "Errors are:"
printResponse(errors)

# Print out the returned Info (if any).
infos = response.getInfo()
print "Infos are:"
printResponse(infos)
```

Keywords

system eam runTask, eam.runTask

system.file

File Functions

The following functions give you access to read and write to files.

[In This Section ...](#)

system.file.fileExists

This function is used in [Python Scripting](#).

Description

Checks to see if a file or folder at a given path exists.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.fileExists(filepath)

- Parameters

[String](#) **filepath** - The path of the file or folder to check.

- Returns

[Boolean](#) - True if the file/folder exists, false otherwise.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This basic example shows how the fileExists function is used in its simplest form:  
if system.file.fileExists("C:\\temp_file.txt"):  
    system.gui.messageBox("Yes, the file exists")  
else:  
    system.gui.messageBox("No, it doesn't exist")
```

Code Snippet

```
# This code uses the fileExists function, along with other system.file.* functions, to prompt the user to  
confirm that they want to overwrite an existing file.  
filename = system.file.writeFile("")  
if filename is not None:  
    reallyWrite = 1  
    if system.file.fileExists(filename):  
        overwriteMessage = "File '%s' already exists. Overwrite?"  
        reallyWrite = system.gui.confirm(overwriteMessage % filename)  
    if reallyWrite:  
        system.file.writeFile(filename, "This will be the contents of my new file")
```

Keywords

system file fileExists, file.fileExists

system.file.getTempFile

This function is used in **Python Scripting**.

Description

Creates a new temp file on the host machine with a certain extension, returning the path to the file. The file is marked to be removed when the Java VM exits.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.getTempFile(extension)

- Parameters

[String](#) extension - A file extension, such as ".txt", to append to the end of the temporary file.

- Returns

[String](#) - The path to the newly created temp file.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code writes some data to a temporary file and then opens that file.  
# Assume that the data variable holds the contents of an Excel (xls) file.  
  
filename = system.file.getTempFile("xls")  
system.file.writeFile(filename, data)  
system.net.openURL("file://" + filename)
```

Keywords

system file getTempFile, file.getTempFile

system.file.openFile

This function is used in [Python Scripting](#).

Description

Shows an Open File dialog box, prompting the user to choose a file to open. Returns the path to the file that the user chose, or None if the user canceled the dialog box. An extension can optionally be passed in that sets the filetype filter to that extension.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.openFile([extension], [defaultLocation])

- Parameters

[String](#) extension - A file extension, such as "pdf", to try to open. [optional]

[String](#) defaultLocation - A folder location, such as "C:\MyFiles", to use as the default folder to store in. [optional]

- Returns

[String](#) - The path to the selected file, or Nothing if canceled.

- Scope

Vision Client

Code Examples

Code Snippet - Opening a File

```
# This code prompts the user to open a file of type 'gif'. If None is returned, it means the user canceled the open dialog box.
```

```
path = system.file.openFile('gif')
if path != None:
    # do something with the file
```

Code Snippet - Opening a File and Specifying a Default Location

```
# This code prompts the user to open a file of type 'pdf' from their stored documents folder. If None is returned, it means the user canceled the open dialog box.
```

```
# Note: The computer running this code needs to have network access to the 'fileserver' computer.
path = system.file.openFile('pdf', '\\fileserver\PDF_Storage')
if path != None:
    # do something with the file
```

Keywords

system file openFile, file.openFile

system.file.openFiles

This function is used in [Python Scripting](#).

Description

Shows an Open File dialog box, prompting the user to choose a file or files to open. Returns the paths to the files that the user chooses, or None if the user canceled the dialog box. An extension can optionally be passed in that sets the filetype filter to that extension.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.openFiles([extension], [defaultLocation])

- Parameters

[String](#) extension - A file extension, such as "pdf", to try to open. [optional]

[String](#) defaultLocation - A folder location, such as "C:\MyFiles", to use as the default folder to store in. [optional]

- Returns

[List](#) - The a list of strings representing the paths to the selected files, or None if canceled.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code prompts the user to open files of type 'gif'. If None is returned, it means the user canceled the open dialog box.
```

```
paths = system.file.openFiles('gif')
if paths != None:
    # Do something with the file.
```

Code Snippet

```
# This code prompts the user to open files of type 'pdf' from their stored documents folder.
# If None is returned, it means the user canceled the open dialog box.
# Note: The computer running this code needs to have network access to the 'fileserver' computer.
path = system.file.openFiles('pdf', '\\fileserver\PDF_Storage')
if path != None:
    # Do something with the file.
```

Code Snippet

```
# This code prompts the user to open files of any type and loop through all returned file paths.
```

```
paths = system.file.openFiles()
if len(paths) != 0:
```

```
for path in paths:  
    # Do something with the file.  
    print path
```

Keywords

system file openFiles, file.openFiles

system.file.readFileAsBytes

This function is used in [Python Scripting](#).

Description

Opens the file found at path filename, and reads the entire file. Returns the file as an array of bytes. Commonly this array of bytes is uploaded to a database table with a column of type BLOB (Binary Large OBject). This upload would be done through an INSERT or UPDATE SQL statement run through the system.db.runPrepUpdate function. You could also write the bytes to another file using the system.file.writeFile function, or send the bytes as an email attachment using system.net.sendEmail.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.readFileAsBytes(filepath)

- Parameters

[String](#) filepath - The path of the file to read.

- Returns

[List\[Byte\]](#) - The contents of the file as an array of bytes.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code prompts the user to choose a file. If the user chooses a file, that file is read and  
# uploaded to a database table called Files into a BLOB column called file_data.
```

```
path = system.file.openFile()  
if path != None:  
    bytes = system.file.readFileAsBytes(path)  
    system.db.runPrepUpdate("INSERT INTO Files (file_data) VALUES (?)", [bytes])
```

Keywords

system file readFileAsBytes, file.readFileAsBytes

system.file.readFileAsString

This function is used in [Python Scripting](#).

Description

Opens the file found at path filename, and reads the entire file. Returns the file as a string. Common things to do with this string would be to load it into the text property of a component, upload it to a database table, or save it to another file using system.file.writeFile function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.readFileAsString(filepath, [encoding])

- Parameters

String filepath - The path of the file to read.

String encoding - The character encoding of the file to be read. Will throw an exception if the string does not represent a supported encoding. Common encodings are "UTF-8", "ISO-8859-1" and "US-ASCII". Default is "UTF-8". [optional]

- Returns

String - The contents of the file as a string.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Reading File as String

```
# This code prompts the user to choose a text file. If the user chooses a file, a text area on the screen is set to display the file.
```

```
path = system.file.openFile("txt")
if path != None:
    contents = system.file.readFileAsString(path)
    event.source.parent.getComponent("Text Area").text = contents
```

Keywords

system file readFileAsString, file.readFileAsString

system.file.saveFile

This function is used in [Python Scripting](#).

Description

Prompts the user to save a new file named filename. The optional extension and typeDesc arguments will be used for a file type filter, if any. If the user accepts the save, the path to that file will be returned. If the user cancels the save, None will be returned.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.saveFile(filename)

- Parameters

[String](#) filename - A file name to suggest to the user.

- Returns

[String](#) - The path to the file that the user decided to save to, or None if they canceled.

- Scope

Vision Client

Syntax

system.file.saveFile(filename, [extension], [typeDesc])

- Parameters

[String](#) filename - A file name to suggest to the user.

[String](#) extension - The appropriate file extension, such as "jpeg", for the file. [optional]

[String](#) typeDesc - A description of the extension, such as "JPEG Image". [optional]

- Returns

[String](#) - The path to the file that the user decided to save to, or None if they canceled.

- Scope

Vision Client

Code Examples

Code Snippet - Saving a File

```
# This code prompts the user to save the text in a text area to a file.

path = system.file.saveFile("myfile.txt")
if path is not None:
    system.file.writeFile(path, event.source.parent.getComponent("Text Area").text)
```

Keywords

system file saveFile, file.saveFile

system.file.writeFile

This function is used in [Python Scripting](#).

Description

Writes the given data to the file at file path filename. If the file exists, the append argument determines whether or not it is overwritten (the default) or appended to. The data argument can be either a string or an array of bytes (commonly retrieved from a BLOB in a database or read from another file using system.file.readFileAsBytes).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.file.writeFile(filepath, charData, [append], [encoding])

- Parameters

[String](#) filepath - The path of the file to write to.

[String](#) charData - The character content to write to the file.

[Boolean](#) append - If true, the file will be appended to if it already exists. If false, the file will be overwritten if it exists. The default is false. [optional]

[String](#) encoding - The character encoding of the file to write. Will throw an exception if the string does not represent a supported encoding. Common encodings are "UTF-8", "ISO-8859-1" and "US-ASCII". Default is "UTF-8". [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Syntax

system.file.writeFile(filepath, data, [append], [encoding])

- Parameters

[String](#) filepath - The path of the file to write to.

[List\[byte\]](#) data - The binary content to write to the file.

[Boolean](#) append - If true, the file will be appended to if it already exists. If false, the file will be overwritten if it exists. The default is false. [optional]

[String](#) encoding - The character encoding of the file to write. Will throw an exception if the string does not represent a supported encoding. Common encodings are "UTF-8", "ISO-8859-1" and "US-ASCII". Default is "UTF-8". [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
#This code downloads a BLOB from a database and saves it to a file.

resultSet = system.db.runQuery("SELECT file_data FROM Files WHERE id=12")
if len(resultSet) > 0: # if the query returned anything...
    data = resultSet[0][0] # grab the BLOB at the 0th row and 0th column
    filename = system.file.saveFile("MyDownloadedFile.xyz")
    if filename is not None:
        system.file.writeFile(filename, data)
```

Code Snippet

```
# This code writes the contents of a text area to a file.

data = event.source.parent.getComponent("Text Area").text
filename = system.file.saveFile("MyDownloadedFile.txt")
if filename is not None:
    system.file.writeFile(filename, data)
```

Keywords

system file writeFile, file.writeFile

system.groups

Transaction Group Functions

The following functions give you access to import and remove Transaction Groups.

[In This Section ...](#)

system.groups.loadFromFile

This function is used in [Python Scripting](#).

Description

Loads a transaction group configuration from an xml export, into the specified project (creating the project if necessary). The mode parameter dictates how overwrites occur.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.groups.loadFromFile(filePath, projectName, mode)

- Parameters

String filePath - The path to a valid transaction group xml or csv file.

String projectName - The name of the project to load into.

Integer mode - How duplicates will be handled: 0=Overwrite, 1=Ignore, 2=Replace the existing project with this one.

- Returns

Nothing

- Scope

Gateway

Code Snippet - Loading a Transaction Group Into a Project

```
# Note that backslashes are used in Windows filepaths, but are also escaped in Python. Thus, we use the double-backslashes here.  
path = "C:\\\\Users\\\\user\\\\Desktop\\\\group.xml"  
  
projName = "MyProject"  
  
# Read a Transaction Group from a file and overwrite any preexisting groups that match those in our file.  
system.groups.loadFromFile(path, projName, 0)
```

Keywords

system groups loadFromFile, groups.loadFromFile

system.groups.removeGroups

This function is used in [Python Scripting](#).

Description

Removes the specified groups from the project. The group paths are "Folder/Path/To/GroupName", separated by forward slashes.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.groups.removeGroups(projectName, paths)

- Parameters

[String](#) projectName - The project to remove from. If the project does not exist, throws an `IllegalArgumentException`.

[List\[String\]](#) paths - A list of paths to remove. Group paths are the full path to the resource, separated by forward slashes, e.g., "Folder/Path/To/GroupName".

- Returns

Nothing

- Scope

Gateway

Code Examples

Code Snippet - Removing Transaction Group from Project

```
projName = "MyProject"
groups = [ "Historical/Group1" , "DataSync/Group2" ]

system.groups.removeGroups(projName, groups)
```

Keywords

system groups removeGroups, groups.removeGroups

system.gui

GUI Functions

The following functions allow you to control windows and create popup interfaces.

Constants

```
system.gui.ACCL_NONE = 0
system.gui.ACCL_CONSTANT = 1
system.gui.ACCL_FAST_TO_SLOW = 2
system.gui.ACCL_SLOW_TO_FAST = 3
system.gui.ACCL_EASE = 4
system.gui.COORD_SCREEN = 0
system.gui.COORD_DESIGNER = 1
```

[In This Section ...](#)

system.gui.chooseColor

This function is used in [Python Scripting](#).

Description

Prompts the user to pick a color using the default color-chooser dialog box.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.chooseColor(initialColor, [dialogTitle])

- Parameters

[Color](#) initialColor - A color to use as a starting point in the color choosing popup.

[String](#) dialogTitle - The title for the color choosing popup. Defaults to "Choose Color". [optional]

- Returns

[Color](#) - The new color chosen by the user.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would be placed in the actionPerformed event of a button  
# and would change the background color of the container the button was placed in.  
  
parent = event.source.parent  
newColor = system.gui.chooseColor(parent.background)  
parent.background = newColor
```

Keywords

system gui chooseColor, gui.chooseColor

system.gui.closeDesktop

This function is used in [Python Scripting](#).

Description

Allows you to close any of the open desktops associated with the current Client. See the [Multi-Monitor Clients](#) page for more details about using multiple monitors.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.closeDesktop(handle)

- Parameters

String handle - The handle for the desktop to close. The screen index cast as a string may be used instead of the handle. If omitted, this will default to the primary desktop. Alternatively, the handle "primary" can be used to refer to the primary desktop.

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# The following example closes desktop 2.  
# The handle must be a string.  
system.gui.closeDesktop("2")
```

Code Snippet

```
# The following example closes the desktop named "Left Monitor".  
system.gui.closeDesktop("Left Monitor")
```

Keywords

system gui closeDesktop, gui.closeDesktop

system.gui.color

This function is used in [Python Scripting](#).

Description

Creates a new color object, either by parsing a string or by having the RGB[A] channels specified explicitly. See [toColor](#) to see a list of available color names.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.color(color)

- Parameters

[String](#) color - A string that will be coerced into a color. Can accept many formats, such as "red" or "#FF0000" or "255,0,0".

- Returns

[Color](#) - The newly created color.

- Scope

Vision Client

Syntax

system.gui.color(red, green, blue, [alpha])

- Parameters,

[Integer](#) red - The red component of the color, an integer 0-255.

[Integer](#) green - The green component of the color, an integer 0-255.

[Integer](#) blue - The blue component of the color, an integer 0-255.

[Integer](#) alpha - The alpha component of the color, an integer 0-255. [optional]

- Returns

[Color](#) - The newly created color.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example changes the background color of a component to red.  
  
myComponent = event.source  
myComponent.background = system.gui.color(255,0,0) # turn the component red
```

Keywords

```
system gui color, gui.color
```

system.gui.confirm

This function is used in [Python Scripting](#).

Description

Displays a confirmation dialog box to the user with "Yes" and "No" options, and a custom message.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.confirm(message, [title], [allowCancel])

- Parameters

[String](#) message - The message to show in the confirmation dialog.

[String](#) title - The title for the confirmation dialog. [optional]

[Boolean](#) allowCancel - Show a cancel button in the dialog. [optional]

- Returns

[Boolean](#) - True if the user selected "Yes"; false if the user selected "No". None if the user selected "Cancel".

- Scope

Vision Client

Code Examples

Code Snippet

```
# By using the confirm function in an if statement, we can let the user confirm an action. In this case,
we shut down the plant if the user confirms it, otherwise, we don't do anything.
```

```
if system.gui.confirm("Are you sure you want to shutdown the plant?",
"Really Shutdown?"):
    system.db.runUpdateQuery( "UPDATE ControlTable SET Shutdown=1" )
```

Keywords

system gui confirm, gui.confirm

system.gui.convertPointToScreen

This function is used in [Python Scripting](#).

Description

Converts a pair of coordinates that are relative to the upper-left corner of a component to be relative to the upper-left corner of the entire screen.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.convertPointToScreen(x, y, event)

- Parameters

[Integer](#) x - The x-coordinate, relative to the component that fired the event.

[Integer](#) y - The y-coordinate, relative to the component that fired the event.

[EventObject](#) event - An event object for a component event.

- Returns

[Tuple](#) - A tuple of (x,y) in screen coordinates.

- Scope

Vision Client

Code Examples

Code Snippet - Getting Location of Mouse Click

```
# This example gets the coordinates where the mouse is (from the corner of the monitor) and displays them
# in a label.
# Get the screen coordinates of the pointer and write them to a label.
# For this example, the code was placed on the root container of a window under the mouseClicked event
# handler.
coords = system.gui.convertPointToScreen(event.x, event.y, event)
event.source.getComponent('Label').text = "x: %s y: %s" %(coords[0], coords[1])
```

Keywords

system gui convertPointToScreen, gui.convertPointToScreen

system.gui.createPopupMenu

This function is used in [Python Scripting](#).

Description

Creates a new popup menu, which can then be shown over a component on a mouse event. To use this function, first create a Python sequence whose entries are strings, and another sequence whose entries are function objects. The strings will be the items that are displayed in your popup menu, and when an item is clicked, its corresponding function will be run. Your functions must accept an event object as an argument. See also: [Functions](#).

It is best to have the menu object created only once via an application specific library function. Then, call the show(event) function on both the mousePressed and mouseReleased events on your component. The reason for this is that different operating systems (Windows, Linux, MacOS) differ in when they like to show the popup menu. The show(event) function detects when the right time is to show itself, either on mouse press or release. See the examples for more.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.createPopupMenu(itemNames, itemFunctions)

- Parameters

[List\[String\]](#) itemNames - A list of names to create popup menu items with.

[List\[String\]](#) itemFunctions - A list of functions to match up with the names. Passing in a None object will cause a separator line to appear in the popup menu, and the corresponding string will not be displayed (note that a corresponding string must be supplied, since the number of elements in the itemFunctions parameter must always match the number of elements in the itemNames parameter).

- Returns

[JPopupMenu](#) - The javax.swing.JPopupMenu that was created.

- Scope

Vision Client

Code Examples

Caution: Event Handlers

A popup menu must be created on either the mouse released or mouse pressed event handlers. This function is not appropriate for invoking on the property change event.

Also, the mouse motions that invoke the popup menu are dependent on the operating system and may behave differently depending on which button you press on the mouse. Because of the different popup-trigger settings on different operating systems, the example code may behave differently on Linux or iOS. The way around this is to do the same code in both the mousePressed and mouseReleased events. In order to avoid code duplication, consider placing the code in a custom method.

Code Snippet

```
# This first example demonstrates the fundamentals of making a popup menu. Put the following script in the mouseReleased event of a component.  
# This will only work on Windows - continue on for cross-platform instructions.  
# Right click on the component to see the resulting pop-up menu that is created with this code.  
  
def sayHello(event):  
    system.gui.messageBox("Hello World")  
menu = system.gui.createPopupMenu(["Click Me"], [sayHello])  
menu.show(event)
```

Code Snippet - Adding a Separator

```
# Similar to the first example, example we'll add an additional option, as well as a separator between
# the two options.

def sayHello(event):
    system.gui.messageBox("Hello World")

def sayGoodbye(event):
    system.gui.messageBox("See you later")

menu = system.gui.createPopupMenu(["Say Hi", "Seperator", "Say Goodbye"], [sayHello, None, sayGoodbye])
menu.show(event)
```

Code Snippet

```
# The following code demonstrates how to edit a component's custom property after you right clicked the
# component.
# This code makes use of functions in order to edit the components custom properties.
# The following code should be located in the mouse released event handler.
# Also, there must be custom properties present on the component in order to handle these functions.
# For example, there must be a custom property called 'DatabaseProvider' that takes a string.
if event.button != event.BUTTON1:
    def editDatabaseProvider(event):
        result = system.gui.inputBox("Database Provider",event.source.parent.DatabaseProvider)
        event.source.parent.DatabaseProvider = result

    def editTable(event):
        result = system.gui.inputBox("Table Name",event.source.parent.Table)
        event.source.parent.Table = result

    def editColumn(event):
        result = system.gui.inputBox("Column Name",event.source.parent.Column)
        event.source.parent.Column = result

    def editKeyColumn(event):
        result = system.gui.inputBox("Key Column Name",event.source.parent.KeyColumn)
        event.source.parent.KeyColumn = result

    names = ["Edit DB Provider", "Edit Table Name", "Edit Column Name", "Edit Key Column"]
    functions = [editDatabaseProvider, editTable, editColumn, editKeyColumn]
    menu = system.gui.createPopupMenu(names, functions)
    menu.show(event)
```

Code Snippet

```
# This example shows a nested popup menu, with menus within menus. All menu items call sayHello().
def sayHello(event):
    system.gui.messageBox("Hello World")
    subMenu = [["Click Me 2", "Click Me 3"], [sayHello, sayHello]]
    menu = system.gui.createPopupMenu(["Click Me", "SubMenu"], [sayHello, subMenu])
    menu.show(event)
```

Keywords

system gui createPopupMenu, gui.createPopupMenu

system.gui.desktop

This function is used in [Python Scripting](#).

Description

Allows for invoking system.gui functions on a specific desktop. See the [Multi-Monitor Clients](#) page for more details.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.desktop(handle)

- Parameters

[String](#) handle - The handle for the desktop to use. The screen index cast as a string may be used instead of the handle. If omitted, this will default to the primary desktop. Alternatively, the handle "primary" can be used to refer to the primary desktop.

- Returns

[WindowUtilities](#) - A copy of [system.gui \(WindowUtilities\)](#) that will be relative to the desktop named by the given handle.

- Scope

Vision Client

Code Examples

Code Snippet - Opening Message Box in a Different Desktop

```
# The following example makes a message box appear on the primary desktop,  
# regardless of where the script originates.  
# system.gui.desktop() function requires a handle be passed to it for this example  
# to work properly.  
system.gui.desktop().messageBox("This will appear on the Primary Desktop")
```

Code Snippet - Showing Open Windows in a Specific Desktop

```
# Retrieves a list of open windows in a specific Desktop. This example assumes a desktop with the handle  
"2nd Desktop" exists.  
name = "2nd Desktop"  
# Returns a tuple of open windows in the Desktop named "2nd Desktop".  
windows = system.gui.desktop(name).getOpenedWindows()  
  
# Converts the tuple to a string, and shows the items in a message box.  
system.gui.messageBox(str(windows))
```

Keywords

system gui desktop, gui.desktop

system.gui.errorBox

This function is used in [Python Scripting](#).

Description

Displays an error-style message box to the user.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.errorBox(message, [title])

- Parameters

[String](#) message - The message to display in an error box. Will accept HTML formatting.

[String](#) title - The title for the error box. [optional]

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet - Using Error Box

```
# Turn on compressor #12, but only if the user has the right credentials.

if 'Supervisor' in system.security.getRoles():
    updateQuery = "UPDATE CompressorControl SET running=1 WHERE compNum = 12"
    system.db.runUpdateQuery(updateQuery)
else:
    errorMessage = "Unable to turn on Compressor 12."
    errorMessage += " You don't have proper security privileges."
    system.gui.errorBox(errorMessage)
```

Keywords

system gui errorBox, gui.errorBox

system.gui.findWindow

This function is used in [Python Scripting](#).

Description

Finds and returns a list of windows with the given path. If the window is not open, an empty list will be returned. Useful for finding all instances of an open window that were opened with [system.nav.openWindowInstance](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.findWindow(path)

- Parameters

String path - The path of the window to search for.

- Returns

List - A list of [window](#) objects. May be empty if window is not open, or have more than one entry if multiple windows are open.

- Scope

Vision Client

Code Examples

Code Snippet - Finding a Window and Closing It

```
# This example finds all open instances of the window named "Popup" and closes them all.  
  
allInstances = system.gui.findWindow( "Popup" )  
for window in allInstances:  
    system.nav.closeWindow(window)
```

Keywords

system gui findWindow, gui.findWindow

system.gui.getCurrentDesktop

This function is used in **Python Scripting**.

Description

Returns the handle of the desktop this function was called from. Commonly used with the [system.gui.desktop](#) and [system.nav.desktop](#) functions.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getCurrentDesktop()

- Parameters

Nothing

- Returns

[String](#) - The handle of the current desktop.

- Scope

Vision Client

Code Examples

Code Snippet - Getting a Desktop's Handle

```
# Shows the desktop's handle in a message box.  
system.gui.messageBox("This desktop's handle is: %s" % system.gui.getCurrentDesktop())
```

Keywords

system gui getCurrentDesktop, gui.getCurrentDesktop

system.gui.getScreenIndex

This function is used in [Python Scripting](#).

Description

Returns an integer value representing the current screen index based on the screen from which this function was called.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getScreenIndex()

- Parameters
 - Nothing
- Returns
 - Integer** - The screen from which the function was called.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# Prints an integer representing the screen from which the function was called.  
print system.gui.getScreenIndex()
```

Keywords

system gui getScreenIndex, gui.getScreenIndex

system.gui.getDesktopHandles

This function is used in [Python Scripting](#).

Description

Gets a list of all **secondary** handles of the open desktops associated with the current client. In this case, **secondary** means any desktop frame opened by the original client frame. Example: If the original client opened two new frames ('left client' and 'right client'), then this function would return ['left client', 'right client'].

See the [Multi-Monitor Clients](#) page for more details about using multiple monitors.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getDesktopHandles()

- Parameters
 - Nothing
- Returns
 - [List\[Any\]](#) - A Python list of unicode strings, representing the handle of all secondary desktop frames.
- Scope
 - Vision Client

Code Examples

Code Snippet - Getting Desktop Handles

```
# The following example list all handles (except the main client)
# in the client console (Help -> Diagnostics -> Console).
print system.gui.getDesktopHandles()
```

Code Snippet - Putting Desktop Handles in a Table

```
# Create the header and fetch handle names.
header = ["Desktop Names"]
handleList = system.gui.getDesktopHandles()

# Change the handle name list into a column.
handleColumn = [[name] for name in handleList]

# Display the handle list in a table component.
event.source.parent.getComponent('Handles Table').data = system.dataset.toDataSet(header, handleColumn)
```

Keywords

```
system gui getDesktopHandles, gui.getDesktopHandles
```

system.gui.getOpenedWindowNames

This function is used in [Python Scripting](#).

Description

Finds all of the currently open windows and returns a tuple of their paths.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getOpenedWindowNames()

- Parameters
 - Nothing
- Returns
 - [Tuple](#) - A tuple of strings, representing the path of each window that is open.
- Scope
 - Vision Client

Code Examples

Code Snippet - Printing Open Window Paths

```
# This example prints out into the console the full path for each opened window.

windows = system.gui.getOpenedWindowNames()
print 'There are %d windows open' % len(windows)
for path in windows:
    print path
```

Keywords

system gui getOpenedWindowNames, gui.getOpenedWindowNames

system.gui.getOpenedWindows

This function is used in **Python Scripting**.

Description

Finds all of the currently open windows and returns a tuple of references to them.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getOpenedWindows()

- Parameters
 - Nothing
- Returns
 - [Tuple](#) - A tuple of the opened windows, not their names, but the actual [window](#) objects themselves.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This example prints out the path of each currently opened window to the console.

windows = system.gui.getOpenedWindows()
print 'There are %d windows open' % len(windows)
for window in windows:
    print window.getPath()
```

Keywords

system gui getOpenedWindows, gui.getOpenedWindows

system.gui.getParentWindow

This function is used in [Python Scripting](#).

Description

Finds the parent (enclosing) window for the component that fired an event and returns a reference to it.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getParentWindow(event)

- Parameters

[EventObject](#) event - A component event object.

- Returns

[FPMIWindow](#)- The [window](#) object that contains the component that fired the event.

- Scope

Vision Client

Code Examples

Code Snippet - Getting Window and Changing Its Title

```
# Use this in an event script to change the window's title.
```

```
window = system.gui.getParentWindow(event)
window.title='This is a new title'
```

Keywords

system gui getParentWindow, gui.getParentWindow

system.gui.getQuality

This function is used in [Python Scripting](#).

Description

Returns the data quality for the property of the given component as an integer. This function can be used to check the quality of a Tag binding on a component in the middle of the script so that alternative actions can be taken in the event of device disconnections.

A description of the quality codes can be found on the [Tag Quality and Overlays](#) page.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getQuality(component, propertyName)

- Parameters

JComponent component - The component whose property is being checked.

String propertyName - The name of the property as a string value.

- Returns

Integer - The data quality of the given property as an integer.

- Scope

Vision Client

Code Examples

Code Snippet

```
# The following code checks the quality code on an component. If a quality is anything other than good, a message appears.

# Fetch the quality code from the Value property on a Numeric Label. The Numeric Label in this example is inside the same container as this script.
qualityCode = system.gui.getQuality(event.source.parent.getComponent('Numeric Label'), "value")

# Evaluate the quality code. If a value other than 192 is returned:
if str(qualityCode) == "Good":
    # The quality code is good, so continue working. This example simply shows a message, but could be modified to do something more meaningful
    system.gui.messageBox("The property is showing good quality")
else:
    # ...then show a message informing the user. Using Python's string formatting (%i) to pass the quality code into the message.
    system.gui.messageBox("Operation Aborted \n The associated tag is showing quality code %s \n Please check the device connection" % qualityCode)
```

Keywords

system gui getQuality, gui.getQuality

system.gui.getScreens

This function is used in [Python Scripting](#).

Description

Get a list of all the monitors on the computer this client is open on. Use with [system.gui.setScreenIndex](#) to move the client.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getScreens()

- Parameters
 - Nothing
- Returns
 - [List\[Tuple\[String, Integer, Integer\]\]](#) - A sequence of tuples of the form (index, width, height) for each screen device (monitor) available.
- Scope
 - Vision Client

Code Examples

Code Snippet - Getting Screen Information

```
# This example fetches monitor data and pushes it to a table in the same container.

screens = system.gui.getScreens()
pyData = []
for screen in screens:
    pyData.append([screen[0], screen[1], screen[2]])

# Push data to 'Table'.
event.source.parent.getComponent('Table').data = system.dataset.toDataSet(["screen", "width", "height"], pyData)
```

Keywords

system gui getScreens, gui.getScreens

system.gui.getSibling

This function is used in **Python Scripting**.

Description

Given a component event object, looks up a sibling component. Shortcut for `event.source.parent.getComponent("siblingName")`. If no such sibling is found, the special value None is returned.

Client Permission Restrictions

This scripting function has no **Client Permission** restrictions.

Syntax

system.gui.getSibling(event, name)

- Parameters

`EventObject` event - A component event object.

`String` name - The name of the sibling component.

- Returns

`VisionComponent` - Returns reference to the sibling component. See [VisionComponent](#).

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example gets its sibling Text Field's text, and uses it.

textField = system.gui.getSibling(event, 'TextField (1)')
if textField is None:
    system.gui.errorBox("There is no text field!")
else:
    system.gui.messageBox("You typed: %s" % textField.text)
```

Keywords

system gui getSibling, gui.getSibling

system.gui.getWindow

This function is used in [Python Scripting](#).

Description

Finds a reference to an open window with the given name. Throws a ValueError if the named window is not open or not found.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getWindow(name)

- Parameters

[String](#) name - The path to the window to field.

- Returns

[Window](#) - A reference to the [window](#) object, if it was open.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example gets the window named 'Overview' and then closes it.

try:
    window = system.gui.getWindow('Overview')
    system.gui.closeWindow(window)

except ValueError:
    system.gui.warningBox("The Overview window isn't open")
```

Code Snippet

```
# This example sets a value on a label component in the 'Header' window.

try:
    window = system.gui.getWindow('Header')
    window.getRootContainer().getComponent('Label').text = "Machine 1 Starting"

except ValueError:
    system.gui.warningBox("The Header window isn't open")
```

Keywords

```
system gui getWindow, gui.getWindow
```

system.gui.getWindowNames

This function is used in [Python Scripting](#).

Description

Returns a list of the paths of all windows in the current project, sorted alphabetically.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.getWindowNames()

- Parameters
 - Nothing
- Returns
 - [Tuple](#) - A tuple of strings, representing the path of each window defined in the current project.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This example opens windows that begin with "Motor" and passes in the currently selected motor number.

motor = event.source.parent.number
windows = system.gui.getWindowNames()
for path in windows:
    if name[:5] == "Motor":
        system.gui.openWindow(path, {"motorNumber":motor})
```

Keywords

system gui getWindowNames, gui.getWindowNames

system.gui.inputBox

This function is used in [Python Scripting](#).

Description

Opens up a popup input dialog box. This dialog box will show a prompt message and allow the user to type in a string. When the user is done, they can click **OK** or **Cancel**. If OK is pressed, this function will return with the value that they typed in. If Cancel is pressed, this function will return the value None.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.inputBox(message, defaultText)

- Parameters
 - String** message - The message to display for the input box. Will accept HTML formatting.
 - String** defaultText - The default text to initialize the input box with.
- Returns
 - String** - The string value that was entered in the input box.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# In the mouseClicked event of a label, this allows the user to change the label's text.  
  
txt = system.gui.inputBox("Enter text:", event.source.text)  
if txt != None:  
    event.source.text = txt
```

Keywords

system gui inputBox, gui.inputBox

system.gui.isTouchscreenModeEnabled

This function is used in [Python Scripting](#).

Description

Checks whether or not the running Client's Touch Screen mode is currently enabled.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.isTouchscreenModeEnabled()

- Parameters
 - Nothing
- Returns
 - [Boolean](#) - True if the Client currently has Touch Screen mode activated.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This example is used in the Client Startup script to check if this Client is being run on a touch
# screen computer (judged by an IP address) and set Touch Screen mode.

 ipAddress = system.net.getIpAddress()
 query = "SELECT COUNT(*) FROM touchscreen_computer_ips WHERE ip_address = '%s' "
 isTouchscreen = system.db.runScalarQuery(query %(ipAddress))
 if isTouchscreen and not system.gui.isTouchscreenModeEnabled():
    system.gui.setTouchscreenModeEnabled(1)
```

Keywords

system gui isTouchscreenModeEnabled, gui.isTouchscreenModeEnabled

system.gui.messageBox

This function is used in [Python Scripting](#).

Description

Displays an informational-style message popup box to the user.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.messageBox(message, title)

- Parameters

 String message - The message to display. Will accept html formatting.

 String title - A title for the message box. [optional]

- Returns

 Nothing

- Scope

 Vision Client

Code Examples

Code Snippet

```
# Display the message Hello World! in a message box.  
system.gui.messageBox("Hello World!")
```

Keywords

system gui messagebox, gui.messageBox

system.gui.openDesktop

This function is used in [Python Scripting](#).

Description

Creates an additional Desktop in a new frame. For more details, see the [Multi-Monitor Clients](#) page.



This function accepts [keyword arguments](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.openDesktop ([screen], [handle], [title], [width], [height], [x], [y], [windows])

- Parameters

Integer screen - The screen index of which screen to place the new frame on. If omitted, screen 0 will be used. [optional]

String handle - A name for the desktop. If omitted, the screen index will be used. [optional]

String title - The title for the new frame. If omitted, the index handle will be used. If the handle and title are omitted, the screen index will be used. [optional]

Integer width - The width for the new desktop's frame. If omitted, frame will become maximized on the specified monitor. [optional]

Integer height - The width for the new desktop's frame. If omitted, frame will become maximized on the specified monitor. [optional]

Integer x - The x coordinate for the new desktop's frame. Only used if both width and height are specified. If omitted, defaults to 0. [optional]

Integer y - The y coordinate for the new desktop's frame. Only used if both width and height are specified. If omitted, defaults to 0. [optional]

PySequence windows - A list of window paths to open in the new Desktop frame. If omitted, the desktop will open without any opened windows. [optional]

- Returns

JFrame - A reference to the new [Desktop frame](#) object.

- Scope

Vision Client

Code Examples

Code Snippet

```
# Create a list of window paths to open in the new desktop.  
windowsToOpen = ["Main Windows/Main Window", "Navigation"]  
  
# Creates a new desktop. The desktop will open the windows listed above.  
system.gui.openDesktop(windows=windowsToOpen)
```

Code Snippet

```
# Creates a new desktop on monitor 0 (primary) with only the Overview window open.  
system.gui.openDesktop(screen=0, windows=["Overview"])
```

Code Snippet

```
# Creates a new desktop on monitor 0 (primary) with only the Overview window open.  
# Including a handle gives the new desktop a name. This is useful for using other desktop scripting  
functions  
system.gui.openDesktop(screen=0, handle="Left Monitor", windows=["Overview"])
```

Keywords

system gui openDesktop, gui.openDesktop

system.gui.openDiagnostics

This function is used in [Python Scripting](#).

Description

Opens the Client runtime diagnostics window, which provides information regarding performance, logging, active threads, connection status, and the console. This provides an opportunity to open the diagnostics window in situations where the menu bar in the client is hidden, and the keyboard shortcut can not be used.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.openDiagnostics()

- Parameters
 - Nothing
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

Code Snippet - Opening Diagnostics Window

```
# Opens the diagnostics window in a running Client.  
system.gui.openDiagnostics()
```

Keywords

system gui openDiagnostics, gui.openDiagnostics

system.gui.passwordBox

This function is used in [Python Scripting](#).

Description

Pops up a special input box that uses a password field, so the text isn't echoed back in clear-text to the user. Returns the text they entered, or None if they canceled the dialog box.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.passwordBox(message, [title], [echoChar])

- Parameters

[String](#) message - The message for the password prompt. Will accept HTML formatting.

[String](#) title - A title for the password prompt. [optional]

[String](#) echoChar - A custom echo character. Defaults to: * [optional]

- Returns

[String](#) - The password that was entered, or None if the prompt was canceled.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example prompts a user for a password before opening the 'Admin' screen.

password = system.gui.passwordBox("Please enter the password." )
if password == "open sesame":
    system.nav.openWindow("Admin")
```

Keywords

system gui passwordBox, gui.passwordBox

system.gui.setScreenIndex

This function is used in [Python Scripting](#).

Description

Moves an open client to a specific monitor. Use with [system.gui.getScreens](#) to identify monitors before moving.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.setScreenIndex(index)

- Parameters
 - [Integer index](#) - The new monitor index for this client to move to. 0 based.
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

Code Snippet - Setting a Screen's Index

```
# This example could be used on a startup script to move the Client to a second monitor.  
system.gui.setScreenIndex(1)
```

Keywords

system gui setScreenIndex, gui.setScreenIndex

system.gui.setTouchscreenModeEnabled

This function is used in [Python Scripting](#).

Description

Alters a running Client's Touch Screen mode on the fly.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.setTouchscreenModeEnabled(enabled)

- Parameters
 - Boolean enabled - The new value for Touch Screen mode being enabled.
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

Code Snippet - Enabling Touchscreen Mode

```
# This example could be used on an input heavy window's internalFrameActivated event to remove Touch
Screen mode.

if system.gui.isTouchscreenModeEnabled():
    system.gui.setTouchscreenModeEnabled(False)
```

Keywords

system gui setTouchscreenModeEnabled, gui.setTouchscreenModeEnabled

system.gui.showNumericKeypad

This function is used in [Python Scripting](#).

Description

Displays a modal on-screen numeric keypad, allowing for arbitrary numeric entry using the mouse, or a finger on a touch screen monitor. Returns the number that the user entered.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.showNumericKeypad(initialValue, [fontSize], [usePasswordMode])

- Parameters

Number initialValue - The value to start the on-screen keypad with.

Integer fontSize - The font size to display in the keypad. [optional]

Boolean usePasswordMode - If True, display a * for each digit. [optional]

- Returns

Number - The value that was entered in the keypad.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This function is a holdover for backwards compatibility. Input components now know when the Client is
# in Touch Screen mode and respond accordingly.
# This script would go in the MouseClicked or MousePressed action of a Text field or Numeric Text field.

# For Integer Numeric Text field:
if system.gui.isTouchscreenModeEnabled():
    event.source.intValue = system.gui.showNumericKeypad(event.source.intValue)

# For Double Numeric Text field:
if system.gui.isTouchscreenModeEnabled():
    event.source.doubleValue = system.gui.showNumericKeypad(event.source.doubleValue)

# For Text field:
# Notice the str() and int() functions used to convert the text to a number and
# vice versa.
# str() and int() are built-in Python functions
if system.gui.isTouchscreenModeEnabled():
    event.source.text = str(system.gui.showNumericKeypad(int(event.source.text)))
```

Keywords

system gui showNumericKeypad, gui.showNumericKeypad

system.gui.showTouchscreenKeyboard

This function is used in [Python Scripting](#).

Description

Displays a modal on-screen keyboard, allowing for arbitrary text entry using the mouse or a finger on a touchscreen monitor. Returns the text that the user entered.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.showTouchscreenKeyboard(initialText, [fontSize], [passwordMode])

- Parameters

String initialText - The text to start the on-screen keyboard with.

Integer fontSize - The font size to display in the keyboard. [optional]

Boolean passwordMode - True to activate password mode, where the text entered isn't echoed back clear-text. [optional]

- Returns

String - The text that was entered in the on-screen keyboard.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This function is a holdover for backwards compatibility. Input components now know when the client is
# in Touch Screen mode and respond accordingly.
# This would go in the MouseClicked or MousePressed action of a Text Field or similar component.

if system.gui.isTouchscreenModeEnabled():
    event.source.text = system.gui.showTouchscreenKeyboard(event.source.text)
```

Keywords

system gui showTouchscreenKeyboard, gui.showTouchscreenKeyboard

system.gui.transform

This function is used in **Python Scripting**.

Description

Sets a component's position and size at runtime. Additional arguments for the duration, framesPerSecond, and acceleration of the operation exist for animation. An optional callback argument will be executed when the transformation is complete.

Note: The transformation is performed in Designer coordinate space on components that are centered or have more than two anchors.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.gui.transform(component, [newX], [newY], [newWidth], [newHeight], [duration], [callback], [framesPerSecond], [acceleration], [coordSpace])
```

- Parameters

[JComponent](#) component - The component to move or resize.

[Integer](#) newX - An x-coordinate to move to, relative to the upper-left corner of the component's parent container. [optional]

[Integer](#) newY - A y-coordinate to move to, relative to the upper-left corner of the component's parent container. [optional]

[Integer](#) newWidth - A width for the component. [optional]

[Integer](#) newHeight - A height for the component. [optional]

[Integer](#) duration - A duration over which the transformation will take place. If omitted or 0, the transform will take place immediately. [optional]

[Callable](#) callback - Function to be called when the transformation is complete. [optional]

[Integer](#) framesPerSecond - Frame rate argument which dictates how often the transformation updates over the given duration. The default is 60 frames per second. [optional]

[Integer](#) acceleration - An optional modifier to the acceleration of the transformation over the given duration. See [system.gui constants](#) for valid arguments. [optional]

[Integer](#) coordSpace - The coordinate space to use. When the default screen coordinates are used, the given size and position are absolute, as they appear in the client at runtime. When Designer Coordinates are used, the given size and position are pre-runtime adjusted values, as they would appear in the Designer. See [system.gui constants](#) for valid arguments. [optional]

- Returns

[Animator](#) animation - An object that contains pause(), resume(), and cancel() methods, allowing for a script to interrupt the animation. See [Animator](#).

- Scope

Vision Client

Code Examples

```
# This example changes the size of a component to 100x100.  
# Run this script from the component that will be changed (i.e., on the mouseEntered event).  
  
system.gui.transform(component=event.source, newWidth=100, newHeight=100)
```

```
# This example moves a component to coordinates 0,0 over the course of 1 second.  
# When the animation is complete, the component is moved back to its original position  
# over the course of 2 seconds, slowing in speed as it approaches the end.  
  
component = event.source.parent.getComponent('Text Field')  
origX = component.x  
origY = component.y  
  
system.gui.transform(  
    component,  
    0, 0,  
    duration=1000,  
    callback=lambda: system.gui.transform(  
        component,  
        origX, origY,  
        duration=2000,  
        acceleration=system.gui.ACCL_FAST_TO_SLOW  
    )  
)
```

Keywords

system gui transform, gui.transform

system.gui.warningBox

This function is used in [Python Scripting](#).

Description

Displays a message to the user in a warning style popup dialog.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.gui.warningBox(message, [title])

- Parameters

 String message - The message to display in the warning box. Will accept HTML formatting.

 String title - The title for the warning box. [optional]

- Returns

 Nothing

- Scope

 Vision Client

Code Examples

Code Snippet

```
# This code shows a yellow popup box similar to the system.gui.messageBox function.  
# Start the motor, or, warn the user if in wrong mode.  
runMode = event.source.parent.getPropertyValue('RunMode')  
if runMode == 1: Cannot start the motor in mode #1  
    system.gui.warningBox("Cannot start the motor, current mode is <B>VIEW MODE</B>")  
else:  
    system.db.runUpdateQuery( "UPDATE MotorControl SET MotorRun=1" )
```

Keywords

system gui warningBox, gui.warningBox

system.math

Math Functions

The following functions assist with running statistical analysis.

[In This Section ...](#)

system.math.geometricMean

This function is used in [Python Scripting](#).

Description

Calculates the geometric mean. Geometric mean is a type of average which indicates a typical value in a set of numbers by using the product of values in the set.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.geometricMean(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

Float - The geometric mean, or NaN if the input was empty or null. Because this uses logs to compute the geometric mean, will return NaN if any entries are negative.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Geometric Mean

```
# Create a List of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Prints the resulting value.  
print system.math.geometricMean(values)
```

Keywords

system math geometricMean, math.geometricMean

system.math.kurtosis

This function is used in **Python Scripting**.

Description

Calculates the kurtosis of a sequence of values. Kurtosis measures if data is peaked or flat relative to normal distribution. A set of data with high kurtosis will have distinct peaks near the mean, while a set of data with low kurtosis will have a flat top near the mean. Uniform distribution is typically a flat line.

Returns NaN (Not a Number) if passed an empty sequence measure of whether the data are heavy-tailed or light-tailed of a given distribution.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.kurtosis(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The kurtosis, or NaN if the input was empty or null. Additionally, returns NaN if the values returned fewer than four values.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Kurtosis

```
# Create a List of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Prints the resulting value.  
print system.math.kurtosis(values)
```

Keywords

system math kurtosis, math.kurtosis

system.math.max

This function is used in [Python Scripting](#).

Description

Given a sequence of values, returns the greatest value in the sequence, also known as the "max" value.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.max(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

Float - The maximum value contained in the values parameter, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Getting the Max Value from List

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Prints the resulting value.  
print system.math.max(values)
```

Keywords

system math max, math.max

system.math.mean

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the arithmetic mean (average).

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.mean(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

Float - The arithmetic mean, or NaN if the input was empty or None.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Mean

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Prints the resulting value.  
print system.math.mean(values)
```

Keywords

system math mean, math.mean

system.math.meanDifference

This function is used in [Python Scripting](#).

Description

Given two sequences of values, calculates the mean of the signed difference between both sequences. In other words, returns the absolute difference between the mean values of two different sets of data.

Throws a DimensionMismatchException if the two sequences have different lengths.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.meanDifference(values1, values2)

- Parameters

[List\[Float\]](#) values1 - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

[List\[Float\]](#) values2 - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The mean difference, or NaN if one of the parameters was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - calculating Mean Difference

```
# Create some lists.  
firstList = [3.5, 5.6, 7.8, 7.4, 3.8]  
secondList = [3.5, 5.6, -7.8, 7.4, -3.8]  
  
# Prints the resulting value.  
print system.math.meanDifference(firstList, secondList)
```

Keywords

system math meanDifference, math.meanDifference

system.math.median

This function is used in [Python Scripting](#).

Description

Takes a sequence of values, and returns the median. The median represents the middle value in a set of data.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.median(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

Float - The median, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Median

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Prints the resulting value.  
print system.math.median(values)
```

Keywords

system math median, math.median

system.math.min

This function is used in [Python Scripting](#).

Description

Given a sequence of numerical values, returns the minimum value, also known as the "min" value.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.min(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The minimum value contained within the 'values' parameter, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Getting the Min Value from List

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Prints the resulting value.  
print system.math.min(values)
```

Keywords

system math min, math.min

system.math.mode

This function is used in [Python Scripting](#).

Description

Given a sequence of values, returns the 'mode', or most frequent values.

Returns an empty list if the sequence was empty or None.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.mode(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values.

- Returns

List[Float] - A Java Array (functionally similar to a Python List) of floats representing the most frequent values in the values parameter. If the values parameter was empty, then an empty list will be returned instead.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Getting the Mode from a List of Values

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 7.8]  
  
# Return the most common values.  
modes = system.math.mode(values)  
  
# Print the first item in the result.  
print modes[0]  
  
# Iterate over the results.  
for number in modes:  
    print number
```

Keywords

system math mode, math.mode

system.math.normalize

This function is used in [Python Scripting](#).

Description

Given a sequence of values, normalizes the values. Normalizing data refers to adjusting values measured on different scales and brings them into alignment to allow the comparison of corresponding normalized values. This creates uniformity of values by eliminating the different units of measurement, and to more easily compare data from different places

Returns an empty list if the sequence was empty or None.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.normalize(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values.

- Returns

[List\[Float\]](#) - A Java array (functionally similar to a Python list) of floats representing normalized input, with a mean of 0 and a standard deviation of 1. Returns an empty array if the input was empty or None. If the standard deviation is 0, will return an array of float NaN (Not a Number).

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 7.8]  
  
# Return the most common values.  
normalized = system.math.mode(values)  
  
# Print the first item in the result.  
print normalized[0]  
  
# Iterate over the results.  
for number in normalized:  
    print number
```

Keywords

system math normalize, math.normalize

system.math.percentile

This function is used in [Python Scripting](#).

Description

Given a sequence of numerical values, estimates the percentile of input.

The percentile is a value on a scale that represents a percentage position in a list of data that can be equal to or below that value: i.e., the 25th percentile is a value below which 25% of observable data points may be found.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.percentile(values, percentile)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

Float percentile - The percentile to compute. A float greater than 0 and less than or equal to 100. Will throw an exception if the percentile is out of bounds.

- Returns

Float - An estimate of the requested percentile of the input, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet

```
# Create a list of values.  
values = [3.5, 5.6, 7.4, 3.8]  
  
# Print the 75th percentile.  
print system.math.percentile(values, 75)
```

Keywords

system math percentile, math.percentile

system.math.populationVariance

This function is used in **Python Scripting**.

Description

Given a sequence of values, returns the population variance. Population variance calculates how values in a dataset are spread out from their average value.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.populationVariance(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The population variance, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Population Variance

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.populationVariance(values)
```

Keywords

system math populationVariance, math.populationVariance

system.math.product

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the product of the sequence: the result of multiplying of all values in the sequence together.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.product(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The product of all values in the values parameter, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Multiplying All Values in a List

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.product(values)
```

Keywords

system math product, math.product

system.math.skewness

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the skewness (third central moment). Skewness is a measure of the degree of asymmetry of a distribution of the mean. If skewed to the left, the distribution has a long tail in the negative direction. If skewed to the right, the tail will be skewed in the positive direction.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.skewness(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The skewness of the values parameter, or NaN if values was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Skewness

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.skewness(values)
```

Keywords

system math skewness, math.skewness

system.math.standardDeviation

This function is used in [Python Scripting](#).

Description

Given a sequence of numerical values, calculates the simple standard deviation. Standard deviation is a calculated number for how close, or how far the values of that dataset are in relation to the mean.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.standardDeviation(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The standard deviation of the values parameter, or NaN if the values was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Calculating Standard Deviation

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.standardDeviation(values)
```

Keywords

system math standardDeviation, math.standardDeviation

system.math.sum

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the sum of all values. The sum is the number returned by addition.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.sum(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a Sequence of numerical values will return NaN.

- Returns

Float - The sum of all values in the values parameter, or NaN if values was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet - Summing All Values In List

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.sum(values)
```

Keywords

system math sum, math.sum

system.math.sumDifference

This function is used in [Python Scripting](#).

Description

Given two sequences of values, calculates the sum of the signed difference between both sequences. In other words, the sum and difference between two sets of numbers.

Throws a DimensionMismatchException if the two sequences have different lengths.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.sumDifference(values1, values2)

- Parameters

[List\[Float\]](#) values1 - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

[List\[Float\]](#) values2 - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The sum difference, or NaN if one of the parameters was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet

```
# Create some lists.  
firstList = [3.5, 5.6, 7.8, 7.4, 3.8]  
secondList = [3.5, 5.6, -7.8, 7.4, -3.8]  
  
# Print the resulting value.  
print system.math.sumDifference(firstList, secondList)
```

Keywords

system math sumDifference, math.sumDifference

system.math.sumLog

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the sum of the natural logs.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.sumLog(values)

- Parameters

List[Float] values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

Float - The sum of the natural logs of the input values, or NaN if the input was empty, None, or contains negative numbers.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.sumLog(values)
```

Keywords

system math sumLog, math.sumLog

system.math.sumSquares

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the sum of the squares of all values. Sum squares measures how far individual values are from the mean by calculating how much variation there is in a set of values.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.sumSquares(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The sum of all squares of the 'values' parameter, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.sumSquares(values)
```

Keywords

system math sumSquares, math.sumSquares

system.math.variance

This function is used in [Python Scripting](#).

Description

Given a sequence of values, calculates the variance of all values. Variance measures how far values in a set are spread out from their mean value.

Returns NaN (Not a Number) if passed an empty sequence.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.math.variance(values)

- Parameters

[List\[Float\]](#) values - A sequence of numerical values. Accepts both integers and floats. The sequence may not contain None type values. However, passing a None type object instead of a sequence of numerical values will return NaN.

- Returns

[Float](#) - The variance of all values in the values parameter, or NaN if the input was empty or null.

- Scope

Gateway, Vision Client, Perspective Session

Code Example

Code Snippet

```
# Create a list of values.  
values = [3.5, 5.6, 7.8, 7.4, 3.8]  
  
# Print the resulting value.  
print system.math.variance(values)
```

Keywords

system math variance, math.variance

system.nav

Navigation Functions

The following functions allow you to open and close windows in the client.

[In This Section ...](#)

system.nav.centerWindow

This function is used in [Python Scripting](#).

Description

Given a window path, or a reference to a window itself, it will center the window. The window should be floating and non-maximized. If the window can't be found, this function will do nothing.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.nav.centerWindow(windowPath)`

- Parameters

[String](#) `windowPath` - The path of the window to center.

- Returns

`Nothing`

- Scope

Vision Client

Syntax

`system.nav.centerWindow(window)`

- Parameters

[FPMIWindow](#) `window` - A reference to the window to center.

- Returns

`Nothing`

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example centers the window named 'Overview'.
system.nav.centerWindow('Overview')
```

Keywords

`system nav centerWindow, nav.centerWindow`

system.nav.closeParentWindow

This function is used in [Python Scripting](#).

Description

Closes the parent window given a component event object.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.closeParentWindow(event)

- Parameters

[EventObject](#) event - A component event object. The enclosing window for the component will be closed. Refer to [EventObject](#).

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# When placed in the actionPerformed event of a button,  
# this code closes the window that contained the button.  
system.nav.closeParentWindow(event)
```

Keywords

system nav closeParentWindow, nav.closeParentWindow

system.nav.closeWindow

This function is used in [Python Scripting](#).

Description

Given a window path, or a reference to a window itself, it will close the window. If the window can't be found, this function will do nothing.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.closeWindow(window)

- Parameters

[FPMIWindow](#) **window** - A reference to the window to close. Refer to the [window](#) objects.

- Returns

[Nothing](#)

- Scope

Vision Client

Syntax

system.nav.closeWindow(windowPath)

- Parameters

[String](#) **windowPath** - The path of a window to close.

- Returns

[Nothing](#)

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example gets the window named 'Overview' and then closes it.  
# If the window isn't open, a warning is shown.  
try:  
    window = system.gui.getWindow('Overview')  
    system.nav.closeWindow(window)  
except ValueError:  
    system.gui.warningBox("The Overview window isn't open")
```

Code Snippet

```
# This example closes the window named 'Overview' in one step.  
# If the window isn't open, the call to closeWindow has no effect.  
system.nav.closeWindow('Overview')
```

Keywords

system nav closeWindow, nav.closeWindow

system.nav.desktop

This function is used in [Python Scripting](#).

Description

Allows for invoking [system.nav](#) functions on a specific desktop. See the [Multi-Monitor Clients](#) page for more details.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.desktop([handle])

- Parameters

[String handle](#)- The handle for the desktop to use. May be omitted for the primary desktop. [optional]

- Returns

[INavUtilities](#) - A copy of [system.nav](#) ([INavUtilities](#)) that will alter the desktop named by the given handle.

- Scope

Vision Client

Code Examples

Code Snippet

```
# The following example closes a window at path "Main Windows/Overview" in the primary desktop,  
# regardless of where the script originates from.  
system.nav.desktop().closeWindow("Main Windows/Overview")
```

Code Snippet

```
# Attempts to swap to a window at path "Main Windows/Main Window" on a specific desktop.  
# This example assumes a desktop with the handle "2nd Desktop" is already open.  
system.nav.desktop("2nd Desktop").swapTo("Main Windows/Main Window")
```

Keywords

system nav desktop, nav.desktop

system.nav.getCurrentWindow

This function is used in [Python Scripting](#).

Description

Returns the path of the current "main screen" window, which is defined as the maximized window. With the [typical navigation](#), there is only ever one maximized window at a time.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.getCurrentWindow()

- Parameters

Nothing

- Returns

[String](#) - The path of the current "main screen" window - the maximized window.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code could run in a global timer script.  
# After a 5-minute timeout, navigate back to the home screen.  
if system.util.getInactivitySeconds()>300 and system.nav.getCurrentWindow() != "Home":  
    system.nav.swapTo("Home")
```

Keywords

system nav getCurrentWindow, nav.getCurrentWindow

system.nav.goBack

This function is used in [Python Scripting](#).

Description

When using the typical navigation strategy, this function will navigate back to the previous main screen window.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.goBack()

- Parameters
 - Nothing
- Returns
 - Window - A reference to window that was navigated to. Refer to the list of [window](#) objects.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code would go in a button to move to the previous screen.  
system.nav.goBack()
```

Keywords

system nav goBack, nav.goBack

system.nav.goForward

This function is used in [Python Scripting](#).

Description

When using the typical navigation strategy, this function will navigate "forward" to the last main screen window the user was on when they executed a [system.nav.goBack](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.goForward()

- Parameters

Nothing

- Returns

[Window](#) - A reference to window that was navigated to. Refer to the list of [window objects](#).

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code goes in a button to move to the last screen that used system.nav.goBack().  
system.nav.goForward()
```

Keywords

system nav goForward, nav.goForward

system.nav.goHome

This function is used in **Python Scripting**.

Description

When using the typical navigation strategy, this function will navigate to the "home" window. This is automatically detected as the first main screen window shown in a project.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.goHome()

- Parameters

Nothing

- Returns

[Window](#)- A reference to the home window that was navigated to. Refer to the list of [window](#) objects.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would go in a button to move to the home screen.  
system.nav.goHome()
```

Keywords

system nav goHome, nav.goHome

system.nav.openWindow

This function is used in [Python Scripting](#).

Description

Opens the window with the given path. If the window is already open, brings it to the front. The optional params dictionary contains key:value pairs which will be used to set the target window's root container's dynamic variables.

For instance, if the window that you are opening is named "TankDisplay" and has a dynamic variable in its root container named "TankNumber", then calling system.nav.openWindow("TankDisplay", {"TankNumber": 4}) will open the "TankDisplay" window and set Root Container.TankNumber to four. This is useful for making parameterized windows, that is, windows that are re-used to display information about like pieces of equipment.
See also: [Parameterized Popup Windows](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.openWindow(path, [params])

- Parameters

[String](#) path - The path to the window to open.

[Dictionary\[String, Any\]](#) params - A dictionary of parameters to pass into the window. The keys in the dictionary must match dynamic property names on the target window's root container. The values for each key will be used to set those properties. [optional]

- Returns

[Window](#) - A reference to the opened window. Refer to the list of [window](#) objects.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This is the simplest form of openWindow.  
system.nav.openWindow( "SomeWindowName" )
```

Code Snippet

```
# A more complex example - a setpoint screen for multiple valves that opens centered.  
titleText = "Third Valve Setpoints"  
tankNo = system.nav.openWindow( "ValveSetPts" , { "valveNum":3, "titleText":titleText } )  
system.nav.centerWindow( "ValveSetPts" )
```

Keywords

system nav openWindow, nav.openWindow

system.nav.openWindowInstance

This function is used in **Python Scripting**.

Description

When called in a Vision Client, it operates exactly like [system.nav.openWindow](#), except that if the named window is already open, then an additional instance of the window will be opened. There is no limit to the number of additional instances of a window that you can open.

When called in the Designer, it operates similar to [system.nav.openWindow](#), except that if the named window is already open the function will swap to the opened window. Additional instances will not be opened. A warning is issued indicating why a new instance was not opened.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.openWindowInstance(path, [params])

- Parameters

[String](#) path - The path to the window to open.

[Dictionary\[String, Any\]](#) params - A dictionary of parameters to pass into the window. The keys in the dictionary must match dynamic property names on the target window's root container. The values for each key will be used to set those properties. [optional]

- Returns

[Window](#) - A reference to the opened window. Refer to the list of [window](#) objects.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This example opens three copies of a single HOA popup screen.  
  
system.nav.openWindowInstance("HOA", {machineNum:3})  
system.nav.openWindowInstance("HOA", {machineNum:4})  
system.nav.openWindowInstance("HOA", {machineNum:5})
```

Keywords

system nav openWindowInstance, nav.openWindowInstance

system.nav.swapTo

This function is used in [Python Scripting](#).

Description

Performs a window swap from the current main screen window to the window specified. Swapping means that the opened window will take the place of the closing window - in this case it will be maximized. See also [Navigation Strategies](#).

This function works like [system.nav.swapWindow](#) except that you cannot specify the source for the swap

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.swapTo(path, [params])

- Parameters

[String](#) path - The path of a window to swap to.

[Dictionary\[String, Any\]](#) params - A dictionary of parameters to pass into the window. The keys in the dictionary must match dynamic property names on the target window's root container. The values for each key will be used to set those properties. [optional]

- Returns

[Window](#) - A reference to the swapped-to window. Refer to the list of [window](#) objects.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would go in a button's ActionPerformed event to swap out of the current window and into a
window named MyWindow
system.nav.swapTo("MyWindow")
```

Code Snippet

```
# This code would go in a button's ActionPerformed event to swap out of the current window and into a
window named MyWindow.
# It also looks at the selected value in a dropdown menu and passes that value into the new window.

# MyWindow's Root Container must have a dynamic property named "paramValue"
dropdown = event.source.parent.getComponent("Dropdown")
system.nav.swapTo("MyWindow", {"paramValue":dropdown.selectedValue})
```

Code Snippet

```
#This code cycles through a dictionary of windows. This could be placed on a Client Event Timer Script to
cycle through some windows.
#The below code assumes that each of the windows are in the same folder (named "Main Windows")
#If the windows are in different folders, then the script would need to be modified to prepend the
correct folder name on the last line of code.
```

```
#Build a dictionary of window names without directories.  
windowDict = {"Overview":"Motors", "Motors":"Alarming", "Alarming":"Scripting", "Scripting":"Overview"}  
#Find the current window  
currentWin = system.nav.getCurrentWindow()  
winObj = system.gui.getWindow(currentWin)  
#Find the next window in the dictionary based on the name of the current window (winObj)  
nextWindow = windowDict[winObj.name]  
#Swap to the next window  
system.nav.swapTo("Main Windows/" + nextWindow)
```

Keywords

system nav swapTo, nav.swapTo

system.nav.swapWindow

This function is used in [Python Scripting](#).

Description

Performs a window swap. This means that one window is closed, and another is opened and takes its place - assuming its size, floating state, and maximization state. This gives a seamless transition - one window seems to simply turn into another.

This function works like [system.nav.swapTo](#) except that you can specify the source and destination for the swap

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.nav.swapWindow(swapFromPath, swapToPath, [params])

- Parameters

[String](#) swapFromPath - The path of the window to swap from. Must be a currently open window, otherwise this will act like an openWindow.

[String](#) swapToPath - The name of the window to swap to.

[Dictionary\[String, Any\]](#) params - A dictionary of parameters to pass into the window. The keys in the dictionary must match dynamic property names on the target window's root container. The values for each key will be used to set those properties. [optional]

- Returns

[Window](#) - A reference to the swapped-to window.

- Scope

Vision Client

Syntax

system.nav.swapWindow(event, swapToPath, [params])

- Parameters

[EventObject](#) event - A component event whose enclosing window will be used as the "swap-from" window.

[String](#) swapToPath - The name of the window to swap to.

[Dictionary\[String, Any\]](#) params - A dictionary of parameters to pass into the window. The keys in the dictionary must match dynamic property names on the target window's root container. The values for each key will be used to set those properties. [optional]

- Returns

[Window](#) - A reference to the swapped-to window. Refer to the list of [window](#) objects.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code goes in a button's ActionPerformed event to swap out of the
# window containing the button and into a window named MyWindow
system.nav.swapWindow(event, "MyWindow")
```

Code Snippet

```
# This code swaps from window named WindowA to a window named WindowB  
system.nav.swapWindow("WindowA", "WindowB")
```

Code Snippet

```
# This code swaps from window named WindowA to a window named WindowB.  
# It also looks at the two calendar popup controls and passes the two selected  
# dates to WindowB. WindowB's root Container must have dynamic properties named  
# "startDate" and "endDate".  
date1 = event.source.parent.getComponent("Start Date").date  
date2 = event.source.parent.getComponent("End Date").date  
system.nav.swapWindow("WindowA", "WindowB", {"startDate":date1, "endDate":date2})
```

Keywords

system nav swapWindow, nav.swapWindow

system.net

Net Functions

The following functions give you access to interact with http services.

[In This Section ...](#)

system.net.getExternalIpAddres

This function is used in [Python Scripting](#).

Description

Returns the Client's IP address, as it is detected by the Gateway. This means that this call will communicate with the Gateway, and the Gateway will tell the Client what IP address its incoming traffic is coming from. If you have a client behind a Network Address Translation (NAT) router, then this address will be the Wide Area Network (WAN) address of the router instead of the Local Area Network (LAN) address of the Client, which is what you'd get with [system.net.getIpAddress](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.net.getExternalIpAddres()

- Parameters
 - Nothing
- Returns
 - String** - A text representation of the Client's IP address, as detected by the Gateway.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# Put this script on a navigation button to restrict users from opening a specific page.

ip = system.net.getExternalIpAddres()
# check if this matches the CEO's IP address
if ip == "66.102.7.104":
    system.nav.swapTo("CEO Dashboard")
else:
    system.nav.swapTo("Manager Dashboard")
```

Keywords

system net getExternalIpAddres, net.getExternalIpAddres

system.net.getHostName

This function is used in [Python Scripting](#).

Description

Returns the host name of the computer that the script was ran on. When run in the Gateway scope, returns the Gateway hostname. When run in the Client scope, returns the Client hostname. On Windows, this is typically the "computer name." For example, might return EAST_WING_WORKS TATION or bobs-laptop.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.net.getHostName()

- Parameters
 - Nothing
- Returns
 - [String](#) - The hostname of the local machine.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Put this script on a navigation button to link dedicated machines to specific screens.
comp = system.net.getHostName()
# Check which line this client is tied to.
if comp == "Line1Computer":
    system.nav.swapTo("Line Detail", {"line":1})
elif comp == "Line2Computer":
    system.nav.swapTo("Line Detail", {"line":2})
else:
    system.nav.swapTo("Line Overview")
```

Keywords

system net getHostName, net.getHostName

system.net.getIpAddress

This function is used in **Python Scripting**.

Description

Returns the IP address of the computer that the script was ran on. When run in the Gateway scope, returns the Gateway IP address. When run in the Client scope, returns the Client IP address.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.net.getIpAddress()

- Parameters
 - Nothing
- Returns
 - [String](#) - Returns the IP address of the local machine, as it sees it.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Put this script on a navigation button to link dedicated machines to specific screens.
ip = system.net.getIpAddress()
# Check which line this client is tied to.
if ip == "10.1.10.5":
    system.nav.swapTo("Line Detail", {"line":1})
elif ip == "10.1.10.6":
    system.nav.swapTo("Line Detail", {"line":2})
else:
    system.nav.swapTo("Line Overview")
```

Keywords

system net getIpAddress, net.getIpAddress

system.net.getRemoteServers

This function is used in [Python Scripting](#).

Description

This function returns a list of Gateway network servers that are visible from the local Gateway.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.net.getRemoteServers([runningOnly])

- Parameters

Boolean runningOnly - If set to true, only servers on the Gateway Network that are running will be returned. Servers that have lost contact with the Gateway Network will be filtered out. [optional]

- Returns

List[String, String] - A list of strings representing Gateway Network server IDs.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following creates a list of running servers on the Gateway Network, and shows the list in a message box.

# Collect the list of running servers.
runningServers = system.net.getRemoteServers(True)

# Initialize the start of the message.
serverStatusText = "The following servers are running:\n"
# Add each running server to the message.
for server in runningServers:
    serverStatusText += "%s \n" % server

# Show the message.
system.gui.messageBox(serverStatusText)
```

Keywords

system net getRemoteServers, net.getRemoteServers

system.net.httpClient

This function is used in [Python Scripting](#).

Description

Provides a general use object that can be used to send and receive HTTP requests. The object created by this function is a wrapper around Java's [HttpClient](#) class. Usage requires creating a [JythonHttpClient](#) object with a call to `system.net.httpClient`, then calling a method (such as `get()`, `post()`) on the [JythonHttpClient](#) to actually issue a request.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.net.httpClient([timeout], [bypass_cert_validation], [username], [password], [proxy], [cookie_policy], [redirect_policy], [customizer])
```

- Parameters

[Integer](#) timeout - A value, in milliseconds, to set the client's connect timeout setting to. Defaults to 60000. [optional]

[Boolean](#) bypass_cert_validation - A boolean indicating whether the client should attempt to validate the certificates of remote servers, if connecting via HTTPS/SSL. Defaults to False. [optional]

[String](#) username - A string indicating the username to use for authentication if the remote server requests authentication; specifically, by responding with a [WWW-Authenticate](#) or [Proxy-Authenticate](#) header. Only supports [Basic authentication](#). If `username` is specified but not `password`, an empty string will be used for the password in the Basic Authentication response. Defaults to None. [optional]

[String](#) password - A string indicating the password to use for authentication. Defaults to None. [optional]

[String](#) proxy - The address of a proxy server, which will be used for HTTP and HTTPS traffic. If a port is not specified as part of that address, it will be assumed from the protocol in the URL, i.e. 80/443. Defaults to None. [optional]

[String](#) cookie_policy - A string representing this client's cookie policy. Accepts values "ACCEPT_ALL", "ACCEPT_NONE", and "ACCEPT_ORIGINAL_SERVER". [Defaults to "ACCEPT_ORIGINAL_SERVER"]

[String](#) redirect_policy - A string representing this client's redirect policy. Acceptable values are listed below. Defaults to "Normal". [optional]

- "NEVER" - never allow redirects
- "ALWAYS" - allow redirects
- "NORMAL" - allows redirects, except those that would downgrade to insecure addresses (i.e., HTTPS redirecting to HTTP)

[Callable](#) customizer - A reference to a function. This function will be called with one argument (an instance of [HttpClient.Builder](#)). The function should operate on that builder instance, which allows for customization of the created HTTP client. Defaults to None. [optional]

- Returns

[JythonHttpClient](#) - An object wrapped around an instance of Java's [HttpClient](#) class. The `httpClient` object has methods that can be called to execute HTTP requests against a server. See the panel below for more details.

- Scope

Gateway, Vision Client, Perspective Session

JythonHttpClient

Once a [JythonHttpClient](#) object has been created, it can be used to handle many HTTP requests without needing to create a new client. Individual HTTP requests can be made with the methods detailed below.

JythonHttpClient Methods

Methods

The following methods return either a `system.net.httpClient#Response` object, or a `system.net.httpClient#Promise` object that will eventually resolve to a Response object, if asynchronous. Asynchronous means that the method will be called, but will not block script execution - so multiple asynchronous calls to network services can be made in succession, without each call "waiting" for the result of the previous. Parameters for these functions are documented below.

Method	Description	Return type
<code>.get()</code>	Sends an HTTP GET call, blocking for a response.	Response
<code>.getAsync()</code>	Sends an HTTP GET call without blocking.	Promise
<code>.post()</code>	Sends an HTTP POST call, blocking for a response.	Response
<code>.postAsync()</code>	Sends an HTTP POST call without blocking.	Promise
<code>.put()</code>	Sends an HTTP PUT call, blocking for a response.	Response
<code>.putAsync()</code>	Sends an HTTP PUT call without blocking.	Promise
<code>.delete()</code>	Sends an HTTP DELETE call, blocking for a response.	Response
<code>.deleteAsync()</code>	Sends an HTTP DELETE call without blocking.	Promise
<code>.patch()</code>	Sends an HTTP PATCH call, blocking for a response.	Response
<code>.patchAsync()</code>	Sends an HTTP PATCH call without blocking.	Promise
<code>.head()</code>	Sends an HTTP HEAD call, blocking for a response.	Response
<code>.headAsync()</code>	Sends an HTTP HEAD call without blocking.	Promise
<code>.options()</code>	Sends an HTTP OPTIONS call, blocking for a response.	Response
<code>.optionsAsync()</code>	Sends an HTTP OPTIONS call without blocking.	Promise
<code>.trace()</code>	Sends an HTTP TRACE call, blocking for a response.	Response
<code>.traceAsync()</code>	Sends an HTTP TRACE call, without blocking for a response.	Promise
<code>.request()</code>	Sends an HTTP request, using a verb specified by the <code>method</code> parameter. Use this method in cases where a non-standard verb is required, and you need the call to block.	Response
<code>.requestAsync()</code>	Sends an HTTP request, with a verb specified by the <code>method</code> parameter. Use this method in cases where a non-standard verb is required, and you do not want the call to block.	Promise

Parameters

Parameters in this section can be used by any of the methods above. Exceptions to this rule will be defined on each parameter.

String url - The URL to connect to. [required]

String method - The method to use in the request. [Required. Used by .request() and .requestAsync() only.]

String or Dictionary params - URL parameters to send with the request. Defaults to None. [optional]

- If supplied as a string, will be directly appended to the URL.
- If supplied as a dictionary, key/value pairs will be automatically URL encoded.

String or Dictionary or byte[] data - Data to send in the request. Defaults to None. [optional]

- String data will be sent with a Content-Type of "text/plain; charset=UTF-8", unless a different Content-Type header was specified.
- Dictionaries will be automatically encoded into JSON to send to the target server, with a Content-Type header set to "application/json; charset=UTF-8" unless a different Content-Type header was specified.
- Byte arrays will be streamed directly to the target server, with a Content-Type header of application/octet-stream unless a different Content-Type header was specified.

String file - The path to a file, relative to the HTTP client instance. If specified, and the path is valid, the data in the file will be sent to the remote server. The file attribute overrides any value set in data; only the file's data will be sent. Defaults to None. [optional]

Dictionary headers - A dictionary of HTTP headers to send with the request. Defaults to None. [optional]

String username - Username to add to a Basic Authorization header in the outgoing request. If `username` is specified, but not `password`, the password is encoded as an empty string. Defaults to None. [optional]

String password - Password to add to a Basic Authorization header in the outgoing request. Defaults to None. [optional]

Integer timeout - The read timeout for this request, in milliseconds. Defaults to 60000. [optional]

JythonHttpClient Attributes

This section documents available attributes on the JythonHttpClient object.

Attribute	Description	Return Type
.javaClient	Returns the underlying Java HttpClient instance.	HTTPClient
.cookieManager	Returns a system.net.httpClient#CookieManager , which can be used to get or set cookies on requests from this client, or to override the cookie storage policy of the client.	CookieManager

CookieManager

Each JythonHttpClient instance has an attached CookieManager. This CookieManager can be accessed to retrieve cookies set by external web services, or to set cookies (i.e., for authentication) before a request is made.

CookieManager Methods and Attributes

This section details methods on the CookieManager. Setting the cookie policy is easiest on the initial `system.net.httpClient` call, but the policy on the CookieManager can be overridden with a call to the built-in `setCookiePolicy` method. Policies are defined in the [Java CookiePolicy interface](#).

Methods and Attributes

Method and /or Attribute	Description	Return type
.getCookieStore()	Returns the underlying CookieStore, which can be used to add, remove, or get cookies that have been set by requests from the parent HttpClient instance. See the Java CookieStore interface for more information.	CookieStore
.cookieStore		

.getCookieManager()	Sends an HTTP GET call, blocking for a response.	CookiePolicy
.setCookiePolicy(policy)	Sets a new CookiePolicy. See the Java CookiePolicy interface for more information.	None

```
from java.net import CookiePolicy

client = system.net.httpClient()
manager = client.getCookieManager()
manager.setCookiePolicy(CookiePolicy.ACCEPT_NONE)
```

Response Object

This section documents the Response object, returned by the request methods on the JythonHttpClient object. This object is simply a wrapper for Java's [HTTPResponse](#) object.

Response Methods and Attributes

This section details methods on the Response object.

Methods and Attributes

Method and/or Attribute	Description	Data type
.getBody() .body	Returns the response content directly.	ByteArray
.getJSON([encoding]) .json	Returns the response content as a dictionary, decoded with the encoding specified by the response. The optional encoding parameter can be used to specified how the JSON should be decoded before being mapped into Python objects (dictionary, list, etc). If the response is not valid JSON, an error will be thrown.	Dictionary
.getText([encoding]) .text	Returns the response content, decoded as a string - either with the charset specified by the response (defaulting to UTF-8 if not specified by the remote server), or using the encoding specified in the function call.	String
.getStatusCode() .statusCode	Return the status code of the response object (i.e., 200 or 404).	Integer
.isGood() .good	Returns True if the response was good (i.e., 200) or False if it was a client or server error (status code between 400 and 599).	Boolean
.isClientError()	Returns True if the response was a client error, as in an HTTP 4XX response.	Boolean

<code>.clientError</code>		
<code>.isServerError()</code>	Returns True if the response was a server error, as in an HTTP 5XX response.	Boolean
<code>.serverError</code>		
<code>.getUrl()</code> <code>.url</code>	Returns the URL this Response connected to.	String
<code>.getHeaders()</code> <code>.headers</code>	Returns a case-insensitive "dictionary" of headers present on the response. Values will always be in a list, even if only a single header value was returned.	Dictionary
<code>.getJavaResponse()</code> <code>.javaResponse</code>	Returns the underlying Java HttpResponse behind this Response.	HttpResponse
<code>.getCookieManager()</code> <code>.cookieManager</code>	Returns the CookieManager. See the CookieManager section for more details.	Cookie Manager
<code>.getRequest()</code> <code>.request</code>	Returns a RequestWrapper object, which has details about the original request that was sent to return this response.	RequestWrapper

Promise Object

This section documents the Promise object, which is returned by the asynchronous methods available on the JythonHttpClient object. This object is a wrapper around Java's [CompletableFuture](#) class, and will return some different object once completed with `.get()`.

Promise Methods and Attributes

Method and/or Attribute	Description	Data type
<code>.get([timeout])</code>	Block for timeout until a result is available. The result object can technically be any type, if chaining, but will be a Response object when calling one of the HttpClient methods. If the timeout is met without a result, an exception will be thrown. The timeout, if unspecified, is 60 seconds.	Any
<code>.then(callback)</code>	Allows for chaining, by returning a new Promise which wraps the provided callback. The callback parameter should be a Python function that either accepts two arguments (the result, or an error, either of which can be None) or a single argument, but is able to accept exceptions as well as valid values.	Promise
<code>.handleException(callback)</code>	In the event of an exception in a potential chain of promises, <code>handleException</code> will be called with one argument (the thrown error) and is expected to return a new fallback value for the next step in the promise chain.	Promise

.whenComplete(callback)	Call the provided callback when this promise finishes evaluating. Callback will be called with return value as the first argument, and any thrown error as the second argument. Any return value will be ignored.	None
.cancel()	Attempt to cancel the wrapped Java future. Returns True if the cancellation succeeded.	Boolean
.getFuture() .future	Returns the underlying Java CompletableFuture object that this Promise contains.	CompletableFuture
.isDone() .done	Returns True if the underlying future has completed - regardless of whether it was a good result or exception.	Boolean

RequestWrapper Object

This section documents the RequestWrapper object, which is simply a wrapper around Java's [HttpRequest](#) object. This object can be used to determine details about the request that was originally sent to populate a Response object.

RequestWrapper Methods and Attributes

This section details methods on the RequestWrapper object.

Methods

Method and/or Attribute	Description	Data type
.getUrl() .url	Returns the actual URL that was contacted in the request.	String
.getMethod() .method	Return the HTTP method used in this request; GET, POST, PATCH, etc.	String
.getHeaders() .headers	Returns a case-insensitive "dictionary" of headers present on the request. Values will always be in a list, even if only a single value is present.	Dictionary
.getTimeout() .timeout	Returns the timeout this query was set to, or -1 if the timeout was invalid.	Integer
.getVersion() .version	Returns the HTTP version used for this request; will be either HTTP_1_1 or HTTP_2.	String
.getJavaRequest() .javaRequest	Returns the underlying Java HttpRequest object directly.	HttpRequest

Code Examples

Example

```
# Create the JythonHttpClient.
client = system.net.httpClient()
```

```
# Sent a GET request.  
response = client.get("https://httpbin.org/get", params={"a": 1, "b": 2})  
  
# Validate the response.  
if response.good:  
    # Do something with the response  
    print response.json['args']['a']
```

Example - Waiting for a Response

```
client = system.net.httpClient()  
  
# Send a non-blocking request to an endpoint that will wait 3 seconds.  
promise = client.GetAsync("https://httpbin.org/delay/3", params={"a": 1, "b": 2})  
  
# This will print before we get a response from the endpoint.  
print "doing something while waiting..."  
# do more work here...  
  
# After the work on the previous lines, we can now block and wait for a response.  
response = promise.get()  
if response.good:  
    print response.json['args']['a']
```

Keywords

system nav httpClient, nav.httpClient

system.net.httpDelete

This function is used in [Python Scripting](#).

Description

Performs an HTTP DELETE to the given URL.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax



This function accepts [keyword arguments](#).

system.net.httpDelete(url, [contentType], [connectTimeout], [readTimeout], [username], [password], [headerValues], [bypassCertValidation])

- Parameters

String url - The URL to send the request to.

String contentType - The MIME type used in the HTTP 'Content-type' header. [optional]

Integer connectTimeout - The timeout for connecting to the URL in milliseconds. Default is 10,000. [optional]

Integer readTimeout - The read timeout for the operation in milliseconds. Default is 60,000. [optional]

String username - If specified, the call will attempt to authenticate with basic HTTP authentication. [optional]

String password - The password used for basic HTTP authentication, if the username parameter is also present. [optional]

Dictionary[String, Integer] headerValues - A dictionary of name/value pairs that will be set in the HTTP header. [optional]

Boolean bypassCertValidation - If the target address is an HTTPS address and this parameter is true, the system will bypass all SSL certificate validation. This is not recommended, though is sometimes necessary for self-signed certificates. [optional]

- Returns

String - The content returned for the DELETE operation.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This example attempts to perform a DELETE operation.  
URL = "http://myURL/folder.resource"  
system.net.httpDelete(URL)
```

Keywords

system net httpDelete, net.httpDelete

system.net.httpGet

This function is used in [Python Scripting](#).

Description

Retrieves the document at the given URL using the HTTP GET protocol. The document is returned as a string. For example, if you use the URL of a website, you will get the same thing you would get by going to that website in a browser and using the browser's "View Source" function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.net.httpGet(url, [connectTimeout], [readTimeout], [username], [password], [headerValues], [bypassCertValidation], [useCaches], [throwOnError])
```

- Parameters

[String](#) url - The URL to retrieve.

[Integer](#) connectTimeout - The timeout for connecting to the URL. In milliseconds. Default is 10,000. [optional]

[Integer](#) readTimeout - The read timeout for the get operation. In milliseconds. Default is 60,000. [optional]

[String](#) username - If specified, the call will attempt to authenticate with basic HTTP authentication. [optional]

[String](#) password - The password used for basic HTTP authentication, if the username parameter is also present. [optional]

[Dictionary\[String, Integer\]](#) headerValues - A dictionary of name/value pairs that will be set in the HTTP header. [optional]

[Boolean](#) bypassCertValidation - If the target address is an HTTPS address, and this parameter is true, the system will bypass all SSL certificate validation. This is not recommended, though is sometimes necessary for self-signed certificates. [optional]

[Boolean](#) useCaches - Will cache the information returned by the httpGet call. If using this for something that constantly updates like an RSS feed, it would be better to set this to False. Default is True. [optional]

[Boolean](#) throwOnError - Set to False if you wish to get the error body rather than a Python exception if the GET request returns an error code (non-200 responsive). Default is True. [optional]

- Returns

[String](#) - The content found at the given URL.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

If you are using JSON, check out the [system.util.jsonEncode](#) and [system.util.jsonDecode](#) functions.

Code Snippet

```
# This code would return the source for Google's homepage.  
source = system.net.httpGet("http://www.google.com")  
print source
```

Code Snippet - Getting Weather Information

```

# This code would query NOAA Weather for the temperature in Folsom, CA.
# NOAA data only works in the US.

# get the json weather response from the NOAA.
lat = "38.6524"
lng = "-121.1896"
url = "https://api.weather.gov/points/%s,%s" %(lat, lng)
noaaResponse = system.net.httpGet(url)
noaaJSON = system.util.jsonDecode(noaaResponse)
# Print to see the response.
print noaaJSON.

# Find the forecast URL.
properties = noaaJSON["properties"]
forecastURL = properties["forecast"]

# Get the forecast from NOAA.
forecastResponse = system.net.httpGet(forecastURL)
forecastJSON = system.util.jsonDecode(forecastResponse)
# Print to see the response.
print forecastJSON.

# Print out the forecast in a human readable way.
periods = forecastJSON["properties"]["periods"]
for data in periods:
    print data["name"]
    print str(data["temperature"])+ " °F"
    print data["detailedForecast"]
    print "" # space to separate the periods

```

Keywords

system net httpGet, net.httpGet

system.net.httpPost

This function is used in [Python Scripting](#).

Description

Retrieves the document at the given URL using the HTTP POST protocol. If a parameter dictionary argument is specified, the entries in the dictionary will be encoded in "application/x-www-form-urlencoded" format, and then posted. You can post arbitrary data as well, but you'll need to specify the MIME type. The document is then returned as a string.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.net.httpPost(url, postParams)

- Parameters

[String](#) url - The URL to post to.

[Dictionary\[String, Integer\]](#) postParams - A dictionary of name/value key pairs to use as the post data.

- Returns

[String](#) - The content returned for the POST operation.

- Scope

Gateway, Vision Client, Perspective Session

Syntax

system.net.httpPost(url, [contentType], [postData], [connectTimeout], [readTimeout], [username], [password], headerValues, [bypassCertValidation], [throwOnError])

- Parameters

[String](#) url - The URL to post to.

[String](#) contentType - The MIME type to use in the HTTP "Content-type" header. [optional]

[String](#) postData - The raw data to post via HTTP. [optional]

[Integer](#) connectTimeout - The timeout for connecting to the url. In milliseconds. Default is 10,000. [optional]

[Integer](#) readTimeout - The read timeout for the get operation. In milliseconds. Default is 60,000. [optional]

[String](#) username - If specified, the call will attempt to authenticate with basic HTTP authentication. [optional]

[String](#) password - The password used for basic http authentication, if the username parameter is also present. [optional]

[Dictionary\[String, Integer\]](#) headerValues - A dictionary of name/value pairs that will be set in the http header. [optional]

[Boolean](#) bypassCertValidation - If the target address is an HTTPS address, and this parameter is True, the system will bypass all SSL certificate validation. This is not recommended, though is sometimes necessary for self-signed certificates. [optional]

[Boolean](#) throwOnError - Set to false if you wish to get the error body rather than a Python exception if the POST request returns an error code (non-200 responsive). Default is True. [optional]

- Returns

[String](#) - The content returned for the POST operation.

- Scope

Gateway, Vision Client, Perspective Session

If you are using JSON, check out the [system.util.jsonEncode](#) and [system.util.jsonDecode](#).

Code Examples

Code Snippet

```
# This code posts a name (first and last) to the post testing page at
# "http://www.snee.com/xml/crud/posttest.cgi", and returns the resulting page as a string.
page = system.net.httpPost("http://www.snee.com/xml/crud/posttest.cgi.wasGettingWayTooManyHits",
{"fname":"Billy", "lname":"Bob"})
print page
```

Code Snippet

```
# This code sends an XML message to a hypothetical URL.
message = "<MyMessage><MyElement>here is the element</MyElement></MyMessage>"
system.net.httpPost("http://www.posttome.xyz/posthere", "text/xml", message)
```

Keywords

system net httpPost, net.httpPost

system.net.httpPut

This function is used in **Python Scripting**.

Description

Performs an HTTP PUT to the given URL. Encodes the given dictionary of parameters using "applications/x-www-form-urlencoded" format.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax



This function accepts [keyword arguments](#).

system.net.httpPut(url, [contentType], putData, [connectTimeout], [readTimeout], [username], [password], [headerValues], [bypassCertValidation])

- Parameters

[String](#) url - The URL to put to.

[String](#) contentType - The MIME type used in the HTTP 'Content-type' header. [optional]

[String](#) putData - The raw data to put via HTTP.

[Inegert](#) connectTimeout - The timeout for connecting to the URL in milliseconds. Default is 10,000. [optional]

[Integer](#) readTimeout - The read timeout for the operation in milliseconds. Default is 60,000. [optional]

[String](#) username - If specified, the call will attempt to authenticate with basic HTTP authentication. [optional]

[String](#) password - The password used for basic HTTP authentication, if the username parameter is also present. [optional]

[Dictionary\[String, Integer\]](#) headerValues - A dictionary of name/value pairs that will be set in the HTTP header. [optional]

[Boolean](#) bypassCertValidation - If the target address is an HTTPS address, and this parameter is true, the system will bypass all SSL certificate validation. This is not recommended, though is sometimes necessary for self-signed certificates. [optional]

- Returns

[String](#) - The content returned for the PUT operation.

- Scope

Gateway, Vision Client, Perspective Session

Syntax

system.net.httpPut(url, putParams)

- Parameters

[String](#) url - The URL to send the request to.

[Dictionary\[String, Integer\]](#) putParams - A dictionary of name/value key pairs to use as the put data.

- Returns

[String](#) - The content returned for the PUT operation.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Example - Simple Test

```
# The following example uses a test URL to echo back the data used in the PUT request.  
# Test URL courtesy of:  
# http://stackoverflow.com/questions/5725430/http-test-server-that-accepts-get-post-calls?  
answertab=votes#tab-top  
  
# Specify URL and parameters to pass in the PUT call.  
URL = "http://httpbin.org/put"  
params = {"testkey": "testValue"}  
  
# Make the PUT request and print the results to the console.  
print system.net.httpPut(URL, params)
```

Code Example - Keyword Arguments

```
# This example attempts to authenticate with a username and password, as well as specify a MIME type.  
# The username and password are static in this example, but could easily use other components to allow  
user input  
# or fetch data out of a database instead.  
  
URL = "http://httpbin.org/put"  
params = {"testkey": "testValue"}  
user = "myUser"  
userPass = "password"  
  
# Make the PUT request and print the results to the console  
print system.net.httpPut(URL, params, username = user, password = userPass, contentType = "text/html")
```

Keywords

system net httpPut, net.httpPut

system.net.openURL

This function is used in **Python Scripting**.

Description

Opens the given URL or URI scheme outside of the currently running Client in whatever application the host operating system deems appropriate. For example, the URL:

"http://www.google.com"

will open in the default web browser, whereas this one:

"file:///C:/Report.pdf"

will likely open in Adobe Acrobat. The Windows network-share style path like:

"\\Fileserver\resources\machine_manual.pdf"

will work as well (in Windows).

Caution: Be careful not to use this function in a full-screen client, as launching an external program will break your full-screen exclusive mode.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.net.openURL(url, [useApplet])

- Parameters

String url - The URL to open in a web browser.

Boolean useApplet - If set to true, and the client is running as an Applet, then the browser instance that launched the applet will be used to open the URL. [optional]

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would open a web page.  
system.net.openURL( "http://www.google.com" )
```

Code Snippet

```
# This code would open a PDF document located at C: on the client computer.  
# Note the double backslashes are needed because backslash is the escape character  
# for Python.  
system.net.openURL("file:///C:\\myPDF.pdf")
```

Code Snippet

```
# This code would open a PDF document from a Windows-based file server.  
# Note the double backslashes are needed because backslash is the escape character  
# for Python.  
system.net.openURL( "\\\\"MyServer\"MyDocs\"document.pdf" )
```

Keywords

system net openURL, net.openURL

system.net.sendEmail

This function is used in **Python Scripting**.

Description

Sends an email through the given SMTP server. Note that this email is relayed first through the Gateway; the Client host machine doesn't need network access to the SMTP server.

You can send text messages to cell phones and pagers using email. Contact your cell carrier for details. If you had a Verizon cell phone with phone number (123) 555-8383, for example, your text messaging email address would be: 1235558383@vtext.com. Try it out!

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax



This function accepts [keyword arguments](#).

system.net.sendEmail(smtp, fromAddr, [subject], [body], [html], to, [attachmentNames], [attachmentData], [timeout], [username], [password], [priority], [smtpProfile], [cc], [bcc], [retries], [replyTo])

- Parameters

String smtp - The address of an SMTP server to send the email through, like "mail.example.com". A port can be specified, like "mail.example.com:25". SSL can also be forced, like "mail.example.com:25:tls".

String fromAddr - An email address to have the email come from.

String subject - The subject line for the email. [optional]

String body - The body text of the email. [optional]

Boolean html - A flag indicating whether or not to send the email as an HTML email. Will auto-detect if omitted. [optional]

List[String] to - A list of email addresses to send to.

List[String] attachmentNames - A list of attachment names. Attachment names must have the correct extension for the file type or an error will occur. [optional]

List[Byte] attachmentData - A list of attachment data, in binary format. [optional]

Integer timeout - A timeout for the email, specified in milliseconds. Defaults to 5 minutes (60,000*5). [optional]

String username - If specified, will be used to authenticate with the SMTP host. [optional]

String password - If specified, will be used to authenticate with the SMTP host. [optional]

String priority - Priority for the message, from "1" to "5", with "1" being highest priority. Defaults to "3" (normal) priority. [optional]

String smtpProfile - If specified, the named SMTP profile defined in the Gateway will be used. If this keyword is present, the smtp, username, and password keywords will be ignored. [optional]

List[String] cc - A list of email addresses to carbon copy. Only available if an smtpProfile is used. [optional]

List[String] bcc - A list of email addresses to blind carbon copy. Only available if an smtpProfile is used. [optional]

Integer retries - The number of additional times to retry sending on failure. Defaults to 0. Only available if an smtpProfile is used. [optional]

List[String] replyTo - An list of addresses to have the recipients reply to. If omitted, this defaults to the from address. [optional]

- Returns

Nothing

- Scope

Code Examples**Code Snippet**

```
# This code sends a simple plain-text email to a single recipient, with no attachments.
body = "Hello, this is an email."
recipients = ["bobsmith@mycompany.com"]
system.net.sendEmail("mail.mycompany.com",
"myemail@mycompany.com", "Here is the email!", body, 0, recipients)
```

Code Snippet

```
# This code sends an HTML-formatted email to multiple recipients (including
# cellphones) with no attachments.
body = "<HTML><BODY><H1>This is a big header</H1>
body += "And this text is <font color='red'>red</font></BODY></HTML>"
recipients = ["bobsmith@mycompany.com", "1235558383@vtext.com", "sally@acme.org", "1235557272@vtext.com"]
myuser = "mycompany"
mypass = "1234"
system.net.sendEmail(smtp="mail.mycompany.com", fromAddr="myemail@mycompany.com",
subject="Here is the email!", body=body, html=1, to=recipients, username=myuser, password=mypass)
```

Code Snippet

```
# This code asks the user for an attachment file and attaches the file.
filePath = system.file.openFile()
if filePath != None:
    # This gets the filename without the C:\folder stuff
    fileName = filePath.split("\\")[-1]
    fileData = system.file.readFileAsBytes(filePath)
    smtp = "mail.mycompany.com"
    sender = "myemail@mycompany.com"
    subject = "Here is the file you requested"
    body = "Hello, this is an email."
    recipients = ["bobsmith@mycompany.com"]
    system.net.sendEmail(smtp, sender, subject, body, 0, recipients, [fileName], [fileData])
```

Code Snippet

```
# This code sends an HTML-formatted email to multiple recipients, including a cc, with no attachments,
# using an smtp server defined in the Gateway.
body = "<HTML><BODY><H1>This is a big header</H1>
body += "And this text is <font color='red'>red</font></BODY></HTML>"
recipients = ["bobsmith@mycompany.com", "1235558383@vtext.com", "sally@acme.org", "1235557272@vtext.com"]
cc_recipients = ["annejones@mycompany.com"]
smtp_server = "mySmtpServer"
system.net.sendEmail(smtpProfile=smtp_server, fromAddr="myemail@mycompany.com", subject="Here is the
email!", body=body, html=1, to=recipients, cc=cc_recipients)
```

Keywords

system net sendEmail, net.sendEmail

system.opc

OPC Functions

The following functions allow you to read, write and browser OPC servers.

[In This Section ...](#)

system.opc.browse

This function is used in [Python Scripting](#).

Description

Allows browsing of the OPC servers in the runtime, returning a list of Tags.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax



This function accepts [keyword arguments](#).

system.opc/browse(opcServer, device, folderPath, opcItemPath)

- Parameters

[String](#) opcServer - The name of the OPC server to browse.

[String](#) device - The name of the device to browse.

[String](#) folderPath - Filters on a folder path. Use * as a wildcard for any number of characters and a ? for a single character.

[String](#) opcItemPath - Filters on a OPC item path. Use * as a wildcard for any number of characters and a ? for a single character.

- Returns

[List\[OPCBrowseTag\]](#) - An array of [OPCBrowseTag](#) objects. OPCBrowseTag has the following functions: getOpcServer(), getOpcItemPath(), getType(), getDisplayName(), getDisplayPath(), getDataType().

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Browse every OPC server.

tags = system.opc.browse()
for row in tags:
    print row.getOpcServer(), row.getOpcItemPath(), row.getType(),
    print row.getDisplayName(), row.getDisplayPath(), row.getDataType()
```

Code Snippet

```
# Example 2: Browse Ignition OPC UA.

tags = system.opc.browse(opcServer="Ignition OPC UA Server")
```

Code Snippet

```
# Example 3: Browse Specific Device.

server = "Ignition OPC UA Server"
tags = system.opc.browse(opcServer=server, device="Dairy Demo Simulator")
```

Code Snippet

```
# Example 4: Browse Specific Folder Path (not OPC item path).

server = "Ignition OPC-UA Server"
tags = system.opc.browse(opcServer=server, folderPath="*Overview/AU 1")
```

Keywords

system opc browse, opc.browse

system.opc.browseServer

This function is used in **Python Scripting**.

Description

When called from a Vision Client, Perspective Session, or the Designer, returns a list of OPCBrowseElement objects for the given server. Otherwise returns a list of PyOPCTagEx objects.

Object Summary

The OPCBrowseElement object has the following methods:

- `getDisplayName()` - Returns the display name of the object.
- `getElementType()` - Returns the element type. Element types are server, device, view, folder, object, datavariable, property and method.
- `getNodeId()` - Returns a string representing the server node ID.

This feature is new in Ignition version **8.1.1**.
[Click here](#) to check out the other new features

- `getDataType()` - Returns data type information.

The PyOPCTagEx object has the following methods to retrieve information:

- `getDisplayName()` - Returns the display name of the object.
- `getElementType()` - Returns the element type. Element types are server, device, view, folder, object, datavariable, property and method.
- `getServerName()` - Returns the server name as a string.

This feature is new in Ignition version **8.1.1**.
[Click here](#) to check out the other new features

- `getDataType()` - Returns data type information.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.browseServer(opcServer, nodeId)

- Parameters

 String `opcServer` - The name of the OPC server connection.

 String `nodeId` - The node ID to browse.

- Returns

 List - A list of [PyOPCTagEx](#) objects.

- Scope

 Gateway, Perspective Session

Syntax - Vision Client Scope

system.opc.browseServer(opcServer, nodeId)

- Parameters

String opcServer - The name of the OPC server connection.

String nodeId - The node ID to browse.

- Returns

List - A list of [OPCBrowseElement](#) objects.

- Scope

Vision Client

Code Examples

Code Snippet

```
# Print the name of all devices on Ignition OPC UA.  
opcServer="Ignition OPC UA Server"  
nodeId = "Devices"  
devices = system.opc.browseServer(opcServer, nodeId)  
for device in devices:  
    print device.getDisplayName()
```

Recursive Browse

```
# This example attempts to recursively browse OPC nodes. Be mindful of the maxDepth in larger systems.  
# The example uses system.util.getLogger asynchronously, so if you're calling this in the Script Console,  
# the output may appear in a different console (i.e., Designer console).  
  
from functools import partial  
  
maxDepth = 1          # Determines how deep the browse will go  
serverName = 'Ignition OPC UA Server'  
myLogger =      system.util.getLogger('My Browse') # Creating a logger to print the results  
  
# Determines where the browse should start. An empty string will start at the root.  
# Alternatively, '[device name]' will start at a certain device.  
root = ''  
  
def browse(nodeId, depth = 0):  
    children = system.opc.browseServer(serverName, nodeId)  
  
    for child in children:  
        elementType = str(child.getElementType())  
        childNodeId = child.getServerNodeId().getNodeId()  
  
        msg = 'Depth - %s, Node - %s' % (depth, childNodeId)  
        myLogger.info(msg)  
  
        # If the element is a folder, try to browse deeper.  
        if (elementType == 'FOLDER' and depth < maxDepth):  
            browse(childNodeId, depth + 1)  
  
system.util.invokeAsynchronous(partial(browse, root))
```

Keywords

system opc browseServer, opc.browseServer

system.opc.browseSimple

This function is used in **Python Scripting**.

Description

Allows browsing of OPC servers in the runtime returning a list of tags. `browseSimple()` takes mandatory parameters, which can be null, while `browse()` uses keyword-style arguments.

Note: The spelling on the `opcServer` and `device` parameters must be exact.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.browseSimple(opcServer, device, folderPath, opcItemPath)

- Parameters

`String` `opcServer` - The name of the OPC server to browse.

`String` `device` - The name of the device to browse.

`String` `folderPath` - Filters on a folder path. Use * as a wildcard for any number of characters and a ? for a single character.

`String` `opcItemPath` - Filters on a OPC item path. Use * as a wildcard for any number of characters and a ? for a single character.

- Returns

`List[OPCBrowseTag]` - An array of [OPCBrowseTag](#) objects. `OPCBrowseTag` has the following functions: `getOpcServer()`, `getOpcItemPath()`, `getType()`, `getDisplayName()`, `getDisplayPath()`, `getDataType()`.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example prints out the the OPC item path for each item in a specific folder.

# Browse Ignition's OPC UA Server. This can be changed to match any connected OPC server.
server = "Ignition OPC UA Server"

# Focus on the "SLC" device connection. This must match a valid device connection in the OPC server.
device = "SLC"

# Specify that the folder path should contain "B3".
FolderPath = "*B3*"

# This example is not filtering on a specific OPCItemPath, so it pass Python's None for this parameter
opcItemPath = None

# Call browseSimple and store the results in a variable. Note that it may take some time to complete the
# browse.
OpcObjects = system.opc.browseSimple(server, device,FolderPath,opcItemPath)
```

```
# For each returned address, print out the OPC item path.  
for address in OpcObjects:  
    print address.getOpcItemPath()
```

Keywords

system opc browseSimple, opc.browseSimple

system.opc.getServer

This function is used in [Python Scripting](#).

Description

Returns a list of server names.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.getServer([includeDisabled])

- Parameters

[Boolean](#) includeDisabled - If set to True, enabled and disabled servers will be returned. If set to False, only enabled servers will be returned. Defaults to False. [optional]

- Returns

[List](#) - A list of server name strings. If no servers are found, returns an empty list.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Printing Ignition OPC UA Servers

```
# Print a list of all server names found.  
servers = system.opc.getServer()  
if not servers:  
    print "No servers found"  
else:  
    for server in servers:  
        print server
```

Keywords

system opc getServer, opc.getServer

system.opc.getServerState

This function is used in [Python Scripting](#).

Description

Retrieves the current state of the given OPC server connection. If the given server is not found, the return value will be None. Otherwise, the return value will be one of these strings:

- UNKNOWN
- FAULTED
- CONNECTING
- CLOSED
- CONNECTED
- DISABLED

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.getServerState(opcServer)

- Parameters

String opcServer - The name of an OPC server connection.

- Returns

String - A string representing the current state of the connection, or None if the connection doesn't exist.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following checks the state of all configured servers and shows them in a message box.  
# This code interacts in the Client scope, so it should be placed on a component, such as a Button.  
  
# Retrieve a list of all servers in the Gateway.  
allServers = system.opc.getServers()  
  
# Initialize a message. The example will append the state of each server to this message.  
# The "\n" at the end of the string adds a new line  
message = "Server State:\n"  
  
# Iterate through each server.  
for server in allServers:  
  
    # For each server, append the server name, a colon, the state of the server, and a new line.  
    message += server + ":" + system.opc.getServerState(server) + "\n"  
  
# Show the state of the servers in a message box.  
system.gui.messageBox(message)
```

Keywords

system opc getServerState, opc.getServerState

system.opc.isServerEnabled

This function is used in [Python Scripting](#).

Description

Checks if an OPC server connection is enabled or disabled.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.isServerEnabled(serverName)

- Parameters

[String](#) **serverName** - The name of an OPC server connection.

- Returns

[Boolean](#) - True if the connection is enabled; false if the connection is disabled

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following iterates through all configured OPC servers, and check if they are enabled or disabled.  
# This code interacts in the Client scope, so it should be placed on a component, such as a Button.  
  
# Retrieve a list of all servers in the Gateway.  
allServers = system.opc.getServerNames()  
  
# Initialize a message. The example will append the state of each server to this message.  
# The "\n" at the end of the string adds a new line  
message = "Server Status:\n"  
  
# Iterate through each server.  
for server in allServers:  
  
    # For each server, append the server name, a colon, the state of the server, and a new line.  
    # isServerEnabled returns a boolean, but may use the string format specifier (%s)  
    message += "%s : %s \n" % (server, system.opc.isServerEnabled(server))  
  
# Show the state of the servers in a message box.  
system.gui.messageBox(message)
```

Keywords

system opc isServerEnabled, opc.isServerEnabled

system.opc.readValue

This function is used in [Python Scripting](#).

Description

Reads a single value directly from an OPC server connection. The address is specified as a string, for example, [MyDevice]N11/N11:0. The object returned from this function has three attributes: value, quality, and timestamp. The value attribute represents the current value for the address specified.

The quality attribute is an OPC UA status code. You can easily check a good quality vs a bad quality by calling the isGood() function on the quality object. The timestamp attribute is Date object that represents the time that the value was retrieved at.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.readValue(opcServer, itemPath)

- Parameters

[String](#) opcServer - The name of the OPC server connection in which the item resides.

[String](#) itemPath - The item path, or address, to read from.

- Returns

[QualifiedValue](#) - A [QualifiedValue](#) object that contains the value, quality, and timestamp returned from the OPC server for the address specified.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
server = "Ignition OPC UA Server"
path = "[SLCSim]_Meta:N7/N7:0"
qualifiedValue = system.opc.readValue(server, path)
print "Value: " + str(qualifiedValue.getValue())
print "Quality: " + qualifiedValue.getQuality().toString()
print "Timestamp: " + qualifiedValue.getTimestamp().toString()
```

Keywords

system opc isServerEnabled, opc.isServerEnabled

system.opc.readValues

This function is used in **Python Scripting**.

Description

This function is equivalent to the system.opc.readValue function, except that it can operate in bulk. You can specify a list of multiple addresses to read from, and you will receive a list of the same length, where each entry is the qualified value object for the corresponding address.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opc.readValues(opcServer, itemPaths)

- Parameters

[String](#) opcServer - The name of the OPC server connection in which the items reside.

[List\[String\]](#) itemPaths - A list of strings, each representing an item path, or address to read from.

- Returns

[List\[QualifiedValue\]](#) - A sequence of [QualifiedValue](#) objects, one for each address specified, in order. Each object will contain the value, quality, and timestamp returned from the OPC server for the corresponding address.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no examples for with this scripting function.

Keywords

system opc readValues, opc.readValues

system.opc.setServerEnabled

This function is used in **Python Scripting**.

Description

Enables or disables an OPC server connection.

Client Permission Restrictions

Permission Type: OPC Server Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.opc.setServerEnabled(serverName, enabled)

- Parameters

 String **serverName**- The name of an OPC server connection.

 Boolean **enabled** - The new state the connection should be set to: true to enable the connection, false to disable.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following will iterate through all configured OPC servers, and check if they are enabled or
disabled.
# If a OPC server is disabled, the code will enable it with a call to setServerEnabled.
# This code interacts in the Client scope, so it should be placed on a component, such as a Button.

# Retrieve a list of all servers in the Gateway.
allServers = system.opc.getServers(True)

# Initialize a message. The empty string is initially used so that the value may be checked later.
message = ""

# Iterate through each server.
for server in allServers:

    # For each server, call isServerEnabled. Uses Python's "not" operator to check if a false value
    is returned.
    if not system.opc.isServerEnabled(server):

        # If disabled, then enable the server.
        system.opc.setServerEnabled(server, True)

        # Append details about the state change we made to the message variable.
        message += "%s \n" % (server)

# Check to see if any changes were made. If the length (len()) of the message is less than 1 character,
then a change wasn't made.
if len(message) < 1:
```

```
# Notify the user that the code did not make any changes.  
system.gui.messageBox("No servers were modified")  
else:  
  
    # Otherwise, let the user know which servers we enabled.  
    system.gui.messageBox("The following servers were modified:\n" + message)
```

Keywords

system opc setServerEnabled, opc.setServerEnabled

system.opc.writeValue

This function is used in **Python Scripting**.

Description

Writes a value directly through an OPC server connection synchronously. Will return an OPC-UA status code object. You can quickly check if the write succeeded by calling `isGood()` on the return value from this function.

Client Permission Restrictions

Permission Type: OPC Server Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.opc.writeValue(opcServer, itemPath, value)

- Parameters

String `opcServer` - The name of the OPC server connection in which the item resides.

String `itemPath` - The item path, or address, to write to.

Object `value` - The value to write to the OPC item.

- Returns

Quality - The status of the write. Use `returnValue.isGood()` to check if the write succeeded. Refer to the list of [writeValue](#) objects.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Writing to an OPC Value

```
server = "Ignition OPC UA Server"
path = "[SLCSim]_Meta:N7/N7:0"
oldQualifiedValue = system.opc.readValue(server, path)
newValue = oldQualifiedValue.getValue() + 1
returnQuality = system.opc.writeValue(server, path, newValue)
if returnQuality.isGood():
    print "Write was successful"
else:
    print "Write failed"
```

Keywords

system opc writeValue, opc.writeValue

system.opc.writeValues

This function is used in **Python Scripting**.

Description

This function is a bulk version of system.opc.writeValue. It takes a list of addresses and a list of objects, which must be the same length. It will write the corresponding object to the corresponding address in bulk. It will return a list of status codes representing the individual write success or failure for each corresponding address.

Client Permission Restrictions

Permission Type: OPC Server Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.opc.writeValues(opcServer, itemPaths, values)

- Parameters

String opcServer - The name of the OPC server connection in which the items reside.

List[String] itemPaths - A list of item paths, or addresses, to write to.

List[Any] values - A list of values to write to each address specified.

- Returns

List[Quality] - An array of Quality objects, each entry corresponding in order to the addresses specified. Refer to the list of [writeValues](#) objects.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no examples for this scripting function.

Keywords

system opc writeValues, opc.writeValues

system.opchda

OPC HDA Functions

The following functions give you access to interact with the HDA types of OPC servers.

[In This Section ...](#)

system.opchda.browse

This function is used in [Python Scripting](#).

Description

Performs a browse at the given root.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.browse(root)

- Parameters

[String](#) **root** - The root at which to browse. Needs to be a qualified path.

- Returns

[Results](#) - The results of the browse operation from the given root. Refer to the [Results](#) object in the SDK.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda browse, opchda/browse

system.opchda.getAggregates

This function is used in [Python Scripting](#).

Description

Will query the server for aggregates that it supports.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.getAggregates(serverName)

- Parameters

[String](#) **serverName** - The name of the defined [OPC-HDA](#) server to query.

- Returns

[List\[Aggregate\]](#) - A list of supported [Aggregate](#) objects. Each object has 'id', 'name', and 'desc' properties defined.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda getAggregates, opchda.getAggregates

system.opchda.getAttributes

This function is used in [Python Scripting](#).

Description

Queries the given server for the item attributes that are available with [system.opchda.readAttributes\(\)](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.getAttributes(serverName)

- Parameters

[String](#) serverName - The name of the defined OPC-HDA server to query.

- Returns

[List\[AttributeInfo\]](#) - A list of AttributeInfo objects. See the AttributeInfo Methods panel for a listing of available methods.

- Scope

Gateway, Vision Client, Perspective Session

AttributeInfo Methods

method	description	return type
getId()	Returns the ID of the attribute.	Integer
getName()	Returns the name of the attribute.	String
getDesc()	Returns the description of the attribute.	String
getType()	Returns the data type of the attribute.	Datatype

Code Examples

There are no code examples for this function.

Keywords

system opchda getAttributes, opchda.getAttributes

system.opchda.getServers

This function is used in **Python Scripting**.

Description

Returns a list of the OPC-HDA servers configured on the system. This call will return all configured and enabled servers, including those that are not currently connected.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.getServers()

- Parameters
Nothing
- Returns
[List\[String\]](#) - A list of the string names of servers.
- Scope
Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda getServers, opchda.getServers

system.opchda.insert

This function is used in **Python Scripting**.

Description

Insert values on the OPC-HDA server.

Client Permission Restrictions

Permission Type: OPC Server Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.opchda.insert(serverName, itemId, value, date, quality)

- Parameters

String serverName - The name of the defined OPC-HDA server.

String itemId - The item ID on which to perform the operation.

Any value - The value to insert.

Any date - The date to insert.

Integer quality - The quality to insert.

- Returns

QualityCode - The result of the insert. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This inserts the value for May 28th, 2014 at 5:42:33.  
  
date = system.date.getDate(2014, 4, 28)  
datetime = system.date.setTime(date, 5, 42, 33)  
system.opchda.insert("MyHistoryServer", "MyItemId", 42.5, datetime, 192)
```

Keywords

system opchda insert, opchda.insert

system.opchda.insertReplace

This function is used in **Python Scripting**.

Description

Inserts values on the OPC-HDA server, or replaces them if they already exist.

Client Permission Restrictions

Permission Type: OPC Server Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.opchda.insertReplace(serverName, itemId, value, date, quality)

- Parameters

String serverName - The name of the defined OPC-HDA server.

String itemId - The item ID on which to perform the operation.

Any value - The value to insert or replace.

Any date - The date to insert or replace.

Integer quality - The quality to insert or replace.

- Returns

QualityCode - The result of the insert or replace operation. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda insertReplace, opchda.insertReplace

system.opchda.isServerAvailable

This function is used in [Python Scripting](#).

Description

Checks to see if the specified OPC-HDA server is defined, enabled, and connected.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.isServerAvailable()

- Parameters
 - String** `serverName` - The name of the OPC-HDA server to check.
- Returns
 - Boolean** - True if the server is available and can be queried, false if not.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda isServerAvailable, opchda.isServerAvailable

system.opchda.readAttributes

This function is used in **Python Scripting**.

Description

Reads the specified attributes for the given item over a time range. Attributes and their IDs are defined in the OPC-HDA specification, and can be discovered by calling [system.opchda.getAttributes\(\)](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.readAttributes(serverName, itemId, attributeIds, startDate, endDate)

- Parameters

[String](#) serverName - The name of the defined OPC-HDA server to read.

[String](#) itemId - The itemID to retrieve attributes for.

[String](#) attributeIds - The integer IDs of the attributes to read. The attribute ids are defined in the OPC-HDA specification. The attributes can also be obtained by calling [system.opchda.getAttributes\(\)](#). Some servers may not support all attributes.

[String](#) startDate - The starting date/time of the query.

[String](#) endDate - The ending date/time of the query.

- Returns

[List\[ReadResults\]](#) - A list of read results which is one-to-one with the requested attributes. The ReadResult object has a serviceResult quality property that indicates whether the call was successful, and it is itself a list of QualifiedValues.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda readAttributes, opchda.readAttributes

system.opchda.readProcessed

This function is used in [Python Scripting](#).

Description

Reads processed values from the OPC-HDA server. Processed values are calculated values, based on the aggregate function requested for each item. The list of aggregates can be obtained by calling [system.opchda.getAggregates\(\)](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.opchda.readProcessed(serverName, itemIds, startDate, endDate, resampleIntervalMS, aggregates)

- Parameters

[String](#) serverName - The name of the defined OPC-HDA server to read.

[List\[String\]](#) itemIds - A list of item ids to read.

[Any](#) startDate - The starting date/time of the query.

[Any](#) endDate - The ending date/time of the query.

[Integer](#) resampleIntervalMS - The interval, in milliseconds, that each value should cover.

[List\[Integer\]](#) aggregates - A list which should be one-to-one with the item ids requested, specifying the integer id of the aggregation function to use. The aggregation ids are defined in the OPC-HDA specification. The list of aggregates can also be obtained by calling [system.opchda.getAggregates\(\)](#).

- Returns

[List\[ReadResults\]](#) - A list of read results which is one-to-one with the item IDs passed in. The ReadResult object has a 'serviceResult' quality property that indicates whether the call was successful, and is itself a list of QualifiedValues.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda readProcessed, opchda.readProcessed

system.opchda.readRaw

This function is used in [Python Scripting](#).

Description

Reads raw values from the OPC-HDA server.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.opchda.readRaw(serverName, itemIds, startDate, endDate, maxValues, boundingValues)
```

- Parameters

[String](#) serverName - The name of the defined OPC-HDA server to read.

[List](#) itemIds - A list of item ids to read.

[Object](#) startDate - The starting date/time of the query.

[Object](#) endDate - The ending date/time of the query.

[Integer](#) maxValues - The maximum number of values to return. 0 or less means unlimited.

[Boolean](#) boundingValues - A boolean indicating whether or not the "bounding values" should be included in the result set. The bounding values provide a value exactly at the start and end dates, but may be resource-intensive to retrieve.

- Returns

[List\[ReadResults\]](#) - A list of read results which is one-to-one with the item IDs passed in. The ReadResult object has a 'serviceResult' quality property that indicates whether the call was successful, and is itself a list of QualifiedValues.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda readRaw, opchda.readRaw

system.opchda.replace

This function is used in **Python Scripting**.

Description

Replaces values on the OPC-HDA server if the given item ID exists.

Client Permission Restrictions

Permission Type: OPC Server Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.opchda.replace(serverName, itemId, value, date, quality)

- Parameters

String serverName - The name of the defined OPC-HDA server.

String itemId - The item ID to perform the operation on.

Object value - The value to replace.

Object date - The date to replace.

Integer quality - The quality to replace.

- Returns

Integer - The items quality resulting from the operation.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system opchda replace, opchda.replace

system.opcua

OPC - UA Functions

The following functions allow you to interact directly with an OPC UA server.

[In This Section ...](#)

system.opcua.callMethod

This function is used in [Python Scripting](#).

Description

Calls a method in an OPC UA server. To make the most of this function, you'll need to be familiar with methods in the [OPC UA server](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.opcua.callMethod(connectionName, objectId, methodId, inputs)`

- Parameters

[String](#) connectionName - The name of the OPC UA connection to the server that the method resides in.

[String](#) objectId - The NodId of the Object Node the Method is a member of.

[String](#) methodId - The NodId of the Method Node to call.

[List](#) inputs - A list of input values expected by the method.

- Returns

[Tuple](#) - A tuple containing the following:

Index Order	Description
0	Resulting StatusCode for the call
1	A list of StatusCode objects corresponding to each input argument
2	A list of output values.

- Scope

Gateway, Perspective Session

The StatusCode Object

This function returns multiple StatusCode objects. StatusCode is a tuple, containing the following:

Index Order	Description
0	The value of the code
1	The name of the code
2	A description of the code

Code Examples

Code Snippet

```
# Call the Server object's GetMonitoredItems method.  
result = system.opcua.callMethod(  
    "Ignition OPC UA Server",  
    "ns=0;i=2253",
```

```
"ns=0;i=11492",
[1]
)

# Below we print the various elements in the results. The print statements could easily be replaced by
something more useful.

# Prints the StatusCode for the call.
print result[0]

# Prints the list of StatusCodes, one for each input argument passed to system.opcua.callMethod.
print result[1]

# Prints the output values from the call.
print result[2]
```

Keywords

system opcua callMethod, opcua.callMethod

system.perspective

Perspective Functions

The following functions offer various ways to interact with a Perspective session from a Python script.

[In This Section ...](#)

system.perspective.alterLogging

This function is used in **Python Scripting**.

Description

Changes Perspective Session logging attributes and levels. All parameters are optional, with the caveat that at least one of them needs to be used.

Caution:

For the `system.perspective.alterLogging` to work, the following line needs to be added to the `ignition.conf` file, and the Gateway restarted. "X" is the next number in the Java Additional Parameters list in the `ignition.conf` file. Note, this will open potential security holes in the Gateway.

```
wrapper.java.additional.X=-Dperspective.enable-client-logging=true
```

```
# Java Additional Parameters
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#wrapper.java.additional.3=-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8001
```

Syntax

 This function accepts **keyword arguments**.

`system.perspective.alterLogging([remoteLoggingEnabled], [level], [remoteLoggingLevel], [sessionId], [pageId])`

- Parameters

`Boolean remoteLoggingEnabled` - Will enable remote logging if True. Remote logging will send log events from the Session to the Gateway under the `perspective.client` logger if they meet the `remoteLevel` logging level requirement. [optional]

`String level` - The desired Session logging level. Possible values are: all, trace, debug, info, warn, error, fatal, or off. The default is info. [optional]

`String remoteLoggingLevel` - The desired remote logging level. Possible values are: all, trace, debug, info, warn, error, fatal, off. The default is warn. [optional]

`String sessionId` - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the `pageId` parameter must be included in the call. [optional]

`String pageId` - Identifier of the page to target. If omitted, the current Page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Alter the logging level to trace.  
system.perspective.alterLogging(level = 'trace')
```

Keywords

system perspective alterLogging, perspective.alterLogging

system.perspective.closeDock

This function is used in **Python Scripting**.

Description

Closes a docked view.

Syntax

system.perspective.closeDock(id, [sessionId,] [pageId])

- Parameters

String id - The unique, preconfigured dock ID for the docked view. Is specified when a view is assigned as docked for a particular page (in [Page Configuration](#)).

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Closes a docked view with the given dock id.  
system.perspective.closeDock('myDockID')
```

Keywords

system perspective closeDock, perspective.closeDock

system.perspective.closePage

This function is used in **Python Scripting**.

Description

Closes the page with the given page id or the current page if no page id is provided. If a message is provided, it is displayed on the page when the page closes. Otherwise the default message (set in the [Project properties](#)) is displayed.

Syntax

system.perspective.closePage([message], [sessionId], [pageID])

- Parameters

String message - The message to display when the page closes. If omitted, the default message (set in the [Project Properties](#)) is shown. [optional]

String sessionId- Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different Session, the pageld parameter must be included in the call. [optional]

String pageld - Identifier of the page to be closed. If omitted, the current pageld is used. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Closes the page with the given pageId.  
system.perspective.closePage('Your page has been closed.')
```

Keywords

system perspective closePage, perspective.closePage

system.perspective.closePopup

This function is used in **Python Scripting**.

Description

Closes a popup view.

Syntax

system.perspective.closePopup(id, [sessionId], [pageId])

- Parameters

String id - The unique identifier for the the popup, given to the popup when first opened. If given an empty string, then the most recently focused popup will be closed.

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current Page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Closes the popup with the given id.  
system.perspective.closePopup('popup 4')
```

Code Snippet

```
# Closes the last focused popup  
system.perspective.closePopup('')
```

Keywords

system perspective closePopup, perspective.closePopup

system.perspective.closeSession

This function is used in **Python Scripting**.

Description

Closes the Perspective Session with the given sessionID or the current Session if no ID is provided. If a message is provided, it is displayed on the page when the Session closes. Otherwise the default message (set in the [Project properties](#)) is displayed.

Note: In the Perspective App, the user is returned to the launch screen.

Syntax

system.perspective.closeSession([message], [sessionId])

- Parameters

String message - The message to display when the Session closes. If omitted, the default message (set in the [Project properties](#)) is shown. [optional]

String sessionId - Identifier of the Session to be closed. If omitted, the current sessionId is used. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Closes the Session with the given sessionId.  
system.perspective.closeSession('Your Session has ended.', '2e1c98a8-182e-43ce-84e8-a71d441c2cce')
```

Keywords

system perspective closeSession, perspective.closeSession

system.perspective.download

This function is used in **Python Scripting**.

Description

Downloads data from the Gateway to a device running a Session.

Syntax

system.perspective.download(filename, data, [contentType], [sessionId], [pageId])

- Parameters

String filename - Suggested name for the downloaded file.

String data - The data to be downloaded. May be a string, a byte[], or an InputStream. Strings will be written in UTF-8 encoding.

String contentType - Value for the "Content-Type" header, for example: "text/plain; charset=utf-8". [optional]

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Downloads the file "myFile.pdf" (located on the Gateway) to the user running the current Session.  
  
filename = 'myFile.pdf'  
data = system.file.readFileAsBytes('C:\\\\'+filename)  
system.perspective.download(filename, data)
```

Keywords

system perspective download, perspective.download

system.perspective.getProjectInfo

This feature is new in Ignition version **8.1.4**
[Click here](#) to check out the other new features

This function is used in **Python Scripting**.

Description

Returns a dictionary of meta data from a Perspective Project. Exact fields are as follows:

key	value
name	The name of the project.
title	The project title.
description	The project description.
lastModified	Date the project was last modified.
lastModifiedBy	Username that last modified the project.
views	An array with path key for each view in the project.
pageConfigs	An array of objects describing the project's configured pages. Each page configuration object will have two properties: <code>url</code> , and <code>primaryView</code> .

For additional information, refer to [Project Settings](#).

Note: This function accepts [keyword arguments](#).

Syntax

system.perspective.getProjectInfo(projectName)

- Parameters

`String projectName` - The name of the project.

- Returns

`Dictionary [String, Any]` - A dictionary of project meta data.

- Scope

Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system perspective getProjectInfo, perspective.getProjectInfo

system.perspective.getSessionInfo

This function is used in **Python Scripting**.

Description

Returns information about one or more Perspective Sessions. The information returned by this function is a combination of information available on the [Perspective Sessions status page](#) on the Gateway, and some Session props (id and userAgent). Exact fields are as follows:

key	value
userAgent	Information the device running the Session. The exact content returned by this key is based on the the browser/device running the Session.
id	Either the Id of the Session (similar to session.prop.id) or if called from the Designer, returns the Designer's id, as listed on the Designers Status page located on the Gateway.
username	Either the username of the logged in user if authenticated, or "Unauthenticated" if an unauthenticated Session.
project	The name of the project running in the Session.
uptime	The number of milliseconds that the Session instance has been running.
clientAddress	The address of the Session.
lastComm	The number of milliseconds since the last communication from the Gateway.
sessionScope	Where the Session is running. Possible values are: designer, browser, iOS, or Android.
activePages	The number of active pages.
recentBytesSent	The number of bytes last sent by the Session to the Gateway.
totalBytesSent	The total number of bytes sent by the Session to the Gateway.
pageIds	An array of page IDs that are currently open in the Session.

Note: This function accepts [keyword arguments](#).

Syntax

system.perspective.getSessionInfo([usernameFilter], [projectFilter])

- Parameters

[String](#) usernameFilter - A filter based on logged in user. [optional]

[String](#) projectFilter - A filter based on the project name. [optional]

- Returns

[List](#) - A list of objects ([PyJsonObjectAdapter](#)).

- Scope

Perspective Session

Code Examples

Code Snippet

```
# This code counts the number of times a user named "billy" is logged in.  
sessions = system.perspective.getSessionInfo("billy")  
print "Billy has %d sessions" % len(sessions)
```

Code Snippet

```
# This script gets all sessions using the "MyProject" project and displays information about them.  
# Get the Session info.  
projectResults = system.perspective.getSessionInfo(projectFilter="MyProject")  
# Loop through the sessions.  
# Enumerate() gives both the Session object and the index.  
for index, sessionObj in enumerate(projectResults):  
    # Print session info  
    print "Session", index, ": username: ", sessionObj["username"], "uptime: ", sessionObj["uptime"], "  
seconds"
```

Keywords

system perspective getSessionInfo, perspective.getSessionInfo

system.perspective.isAuthorized

This function is used in **Python Scripting**.

Description

Checks if the user in the current Session is authorized against a target collection of security levels.

Syntax

system.perspective.isAuthorized(isAllOf, securityLevels, [sessionId])

- Parameters

Boolean isAllOf - True if the current user must have all of the given security levels to be authorized. False if the current user must have at least one of the given security levels to be authorized.

List[String] securityLevels - An array of string paths to a security level node in the form of "Path/To/Node". Each level in the tree is delimited by a forward slash character. The public node is never a part of the path.

This feature is new in Ignition version **8.1.3**

[Click here](#) to check out the other new features

String sessionId - New in 8.1.3. Identifier of the Session to target. If omitted, the current Session will be used automatically. [optional]

- Returns

True if the user logged into the specified session is authorized, false otherwise.

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Returns true if the current user has either Administrator or Baz.  
# Returns false if they have neither.  
path1 = "Authenticated/Roles/Administrator"  
path2 = "Foo/Bar/Baz"  
isAuthorized = system.perspective.isAuthorized(False, [path1, path2])
```

Keywords

system perspective isAuthorized, perspective.isAuthorized

system.perspective.login

This function is used in [Python Scripting](#).

Description

Triggers a login event that will allow the user to log in with the project's configured Identity Provider (IdP). For this function to work, an IdP must be set in Perspective [Project properties](#). This is particularly useful when you want it to be possible to start a session without authentication and sign in to access certain restricted features.

Caution:

Be advised that this function should not be used in the same script, or in the same triggering event as [system.perspective.logout](#). Logging in and Logging out should always triggered by separate events.

The recommend approach to logging out a user, and then quickly logging in as different user, is to set the `forceAuth` parameter to "True" on the [system.perspective.login](#) function.

Syntax

system.perspective.login([sessionId], [pageId], [forceAuth])

- Parameters

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the Page to target. If omitted, the current Page will be used automatically. [optional]

Boolean forceAuth- Determines if Ignition should ask the IdP to re-authenticate the user, even if the user is already signed into the IdP. If set to true, then the IdP will ask the user to re-enter their credentials. If set to false, then the Gateway will request that the IdP use the last provided credentials for the Session, potentially allowing re-authentication without requiring the user to re-type their credentials. Note that support for this argument is determined by the IdP; the IdP may choose to ignore this request. If this parameter is omitted, then the function will use the re-authentication setting defined under [Project properties](#). [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Force Authentication

```
# When forceAuth is True, the user will always be required to type in their credentials, even if they're already logged in.  
system.perspective.login(forceAuth=True)
```

Keywords

system perspective login, perspective.login

system.perspective.logout

This function is used in **Python Scripting**.

Description

Triggers a logout event, which will log the user out. For this function to work, an Identity Provider (IdP) must be set in the Perspective [Project properties](#).

Caution:

Be advised that this function should not be used in the same script, or in the same triggering event as [system.perspective.login](#). Logging in and Logging out should always triggered by separate events.

The recommend approach to logging out a user, and then quickly logging in as different user, is to set the `forceAuth` parameter to "True" on the [system.perspective.login](#) function.

Syntax

system.perspective.logout([sessionId], [pageId])

- Parameters

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different Session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the Page to target. If omitted, the current Page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system perspective logout, perspective.logout

system.perspective.navigate

This function is used in **Python Scripting**.

Description

Navigate the session to a specified view or mounted page.

The function can be used in three different ways, depending on which parameter is specified:

- **page**: Navigates to a Perspective page.
- **url**: Navigates to a web address, so the function can be used to navigate the user to a web portal, search engine, or other website. This parameter is specified via keyword argument.
- **view**: Navigates to a view. Note that using this parameter does not modify the web browser's address bar, so the browser's history will not contain a listing for the new view. This parameter is specified via keyword argument.

Syntax

Note: This function accepts [keyword arguments](#).

system.perspective.navigate(page, [url], [view], params, [sessionId], [pageId])

- Parameters

[String](#) **page** - The URL of a Perspective page to navigate to.

[String](#) **url** - The URL of a web address to navigate to. If the page or view parameters are specified, then this parameter is ignored. [optional]

[String](#) **view** - If specified, will navigate to a specific view. Navigating to a view with this parameter does not change the address in the web browser. Thus the web browser's back button will not be able to return the user to the previous view. If the page parameter is specified, then this parameter is ignored. [optional]

[Dictionary\[String, String\]](#) **params** - Used only in conjunction with the view parameter, Dictionary of values to pass to any parameters on the view. [optional]

[String](#) **sessionId** - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

[String](#) **pageId** - Identifier of the page to target. If omitted, the current page will be used automatically.

- Returns

[Nothing](#)

- Scope

Perspective Session

Code Examples

Code Snippet - Page

```
# Navigating to a perspective-page. The 'page' parameter doesn't require the use of a keyword argument.  
system.perspective.navigate('/new-page')
```

Code Snippet - Web Address

```
# Navigating to a web address. Note that we're using a keyword argument here.  
# web addresses must use a scheme (like 'http://') at the beginning  
system.perspective.navigate(url = 'http://docs.inductiveautomation.com')
```

Code Snippet - View

```
# Navigating to a new view. Again, we need to use a keyword argument. We are passing in two parameters,
# called "myParam" and "myParam2".
system.perspective.navigate(view = 'folder/myView', params = {'myParam':1,'myParam2':'Test'})
```

Keywords

system perspective navigate, perspective.navigate

system.perspective.navigateBack

This function is used in **Python Scripting**.

This feature is new in Ignition version **8.1.5**

[Click here](#) to check out the other new features

Description

Navigate the session to a specified view or mounted page. This is similar to a browser's "back" function.

Syntax

Note: This function accepts [keyword arguments](#).

system.perspective.navigateBack([sessionId], [pageId])

- Parameters

 String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. [optional]

 String pageId - Identifier of the page to target. If omitted, the current page will be used automatically. [optional]

- Returns

 Nothing

- Scope

 Perspective Session

Code Examples

Code Snippet

```
system.perspective.navigateBack()
```

Keywords

system perspective navigateBack, perspective.navigateBack

system.perspective.navigateForward

This function is used in **Python Scripting**.

This feature is new in Ignition version **8.1.5**

[Click here](#) to check out the other new features

Description

Navigate the session to a specified view or mounted page. This is similar to a browser's "forward" function.

Syntax

Note: This function accepts [keyword arguments](#).

system.perspective.navigateForward([sessionId], [pageId])

- Parameters

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
system.perspective.navigateForward()
```

Keywords

system perspective navigateForward, perspective.navigateForward

system.perspective.openDock

This function is used in **Python Scripting**.

Description

Opens a docked View. Requires the preconfigured dock ID for the view.

Syntax

system.perspective.openDock(id [, sessionId, pageld])

- Parameters

String id - The unique, preconfigured dock ID for the docked View. Is specified when a View is assigned as docked for a particular Page (in Page Configuration).

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different Session, then the pageld parameter must be included in the call. [optional]

String pageld - Identifier of the Page to target. If omitted, the current Page will be used automatically. [optional]

Dictionary[String, String] params - Parameters that can be passed into the docked view. Must match the docked views View Parameters. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet - View

```
# Opens a docked view with a dock ID of "myDockID" on the current page and Session.  
system.perspective.openDock("myDockID", params = {"stationNum":2})
```

Keywords

system perspective openDock, perspective.openDock

system.perspective.openPopup

This function is used in **Python Scripting**.

Description

Open a [popup view](#) over the given page.

Syntax

```
system.perspective.openPopup(id, view, [params], [title], [position], [showCloseIcon], [draggable], [resizable], [modal], [overlayDismiss], [sessionId], [pageId], [viewportBound])
```

- Parameters

String id - A unique popup string. Will be used to close the popup from other popup or script actions.

String view - The path to the View to use in the popup.

Dictionary[String, Any] params - Dictionary of key-value pairs to use as input parameters to the View. Added in 8.0.1. [optional]

String title - Text to display in the title bar. Defaults to an empty string. [optional]

Dictionary[String, Integer] position - Dictionary of key-value pairs to use for position. Possible position keys are: left, top, right, bottom, width, height. Defaults to the center of the window. [optional]

Boolean showCloseIcon - Shows the close icon if True. Defaults to True. [optional]

Boolean draggable - Allows the popup to be dragged if True. Defaults to True. [optional]

Boolean resizable - Allows the popup to be resized if True. Defaults to False. [optional]

Boolean modal - Makes the popup modal if True. A modal popup is the only view the user can interact with. Defaults to False. [optional]

Boolean overlayDismiss - Allows the user to dismiss and close a modal popup by clicking outside of it if True. Defaults to False. [optional]

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. [optional]

String pageId - Identifier of the Page to target. If omitted, the current Page will be used automatically. When targeting a different session, then this parameter must be included in the call. [optional]

This feature is new in Ignition version **8.1.3**
[Click here](#) to check out the other new features

Boolean viewportBound - If True, popups will be "shifted" to always open within the bounds of the viewport. If the popup would be larger than the viewport, then it will be resized to fit within the bounds. Default is False. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Opens a popup view. We are passing in two parameters, called "myParam" and "myParam2". We also set some additional properties of the popup.
```

```
system.perspective.openPopup("myPopupId", 'folder/myView', params = {'myParam':1,'myParam2':'Test'},  
showCloseIcon = False, resizable = True)
```

Code Snippet

```
# Opens a popup view. The top left corner of the popup will be 100 pixels from the left and top edges of  
the session.  
system.perspective.openPopup('myPopupId', 'folder/myView', position = {'left':100,'top':100})
```

Keywords

system perspective openPopup, perspective.openPopup

system.perspective.print

This function is used in **Python Scripting**.

Description

Sends print statements to the Script Console when in the Designer. When in a Session, sends print statements to the Output Console. This function makes scripting diagnostics easier.

Syntax

system.perspective.print([message], [sessionId], [pageId], [destination])

- Parameters

 String message - The print statement that will be displayed on the console. [optional]

 String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

 String pageId - Identifier of the page to target. If omitted, the current Page will be used automatically. [optional]

 String destination - Where the message should be printed. If specified, must be "client", "gateway", or "all". Default is "client". [optional]

- Returns

 Nothing

- Scope

 Perspective Session

Code Examples

Code Snippet - View

```
# Sends print statement to the console.  
system.perspective.print(message="Hello World", destination="gateway")
```

Keywords

system perspective print, perspective.print

system.perspective.refresh

This function is used in **Python Scripting**.

Description

Triggers a refresh of the page.

Note: This method should not be confused with the [refreshBinding](#) component method, which automatically fires a binding on a Perspective component property.

Syntax

system.perspective.refresh([sessionId], [pageId])

- Parameters

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current Page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system perspective refresh, perspective.refresh

system.perspective.sendMessage

This function is used in **Python Scripting**.

Description

Send a message to a message handler within the same session.

The Scope Parameter

It is important to be mindful of the scope parameter when calling this function. It is possible to have multiple instances of a view open in a single page, thus invoking the function with a value of "page" for the scope parameter (or omitting the parameter) will invoke the message handlers on all valid message types. This advice is also applicable when the scope parameter value is passed as "session", as all instances of the matching message type in the whole session will be called.

Syntax

system.perspective.sendMessage(messageType, payload, [scope], [sessionId], [pageId])

- Parameters

String messageType - The message type that will be invoked. Message handlers configured within the project are listening for messages of a specific messageType.

Dictionary[String, String] payload - A Python dictionary representing any parameters that will be passed to the message handler.

String scope - The scope that the message should be delivered to. Valid values are "session", "page", or "view". If omitted, "page" will be used. [optional]

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current page will be used. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet

```
# Sends a message to all Message Handlers configured on the current view, indicating that a new item has been added to a list.  
system.perspective.sendMessage("NewItem", payload = { "itemName": "banana", "itemQuantity": 6 }, scope = "view")
```

Keywords

system perspective sendMessage, perspective.sendMessage

system.perspective.setTheme

This function is used in **Python Scripting**.

Description

Changes the theme in a page to the specified theme.

Note that this function only changes the theme for a single page, not the entire session. To change the theme for a session, write directly to the `session.theme` property instead.

Syntax

`system.perspective.setTheme(name, [sessionId], [pageId])`

- Parameters

`String name` - The theme name to switch to. Possible values are "dark" or "light".

`String sessionId` - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the `pageId` parameter must be included in the call. [optional]

`String pageId` - Identifier of the page to target. If omitted, the current page will be used automatically. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet - Changing Session Theme

```
# Change the current page's theme to the dark theme.  
system.perspective.setTheme( "dark" )
```

Keywords

system perspective setTheme, perspective.setTheme

system.perspective.toggleDock

This function is used in **Python Scripting**.

Description

Toggles a docked view.

Syntax

system.perspective.toggleDock(id, [sessionId], [pageId])

- Parameters

String id - The unique, preconfigured 'Dock ID' for the docked view. Is specified when a view is assigned as docked for a particular page (in Page Configuration).

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the Page to target. If omitted, the current Page will be used automatically. [optional]

Dictionary[String, String] params - Parameters that can be passed into the docked view. Must match the docked views View Parameters. [optional]

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet - View

```
# Toggles a docked view with ID "myDockID". We are passing in two parameters, called "myParam" and "myParam2".
system.perspective.toggleDock('myDockID', params = {'myParam':1,'myParam2':'Test'})
```

Keywords

system perspective toggleDock, perspective.toggleDock

system.perspective.togglePopup

This function is used in **Python Scripting**.

Description

Toggles a [popup view](#). Will open up the popup if it has not been opened yet. Otherwise, it will close the currently opened popup.

Syntax

```
system.perspective.togglePopup(id, view, [params], [title], [position], [showCloseIcon], [draggable], [resizable], [modal], [overlayDismiss], [sessionId], [pageId], [viewportBound])
```

- Parameters

String id - A unique popup string. Will be used to close the popup from other popup or script actions..

String view - The path to the view to use in the popup.

Dictionary[String, Any] params - Dictionary of key-value pairs to use as input parameters to the View. [optional]

String title - Text to display in the title bar. Defaults to an empty string. [optional]

Dictionary[String, Integer] position - Dictionary of key-value pairs to use for position. Possible position keys are: left, top, right, bottom, width, height. Defaults to the center of the window. [optional]

Boolean showCloseIcon - Will show the close icon if True. Defaults to True. [optional]

Boolean draggable - Will allow the popup to be dragged if True. Defaults to True. [optional]

Boolean resizable - Will allow the popup to be resized if True. Defaults to False. [optional]

Boolean modal - Will make the popup modal if True. A modal popup is the only view the user can interact with. Defaults to False. [optional]

Boolean overlayDismiss - Will allow the user to dismiss and close a modal popup by clicking outside of it if True. Defaults to False. [optional]

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically. When targeting a different session, then the pageId parameter must be included in the call. [optional]

String pageId - Identifier of the page to target. If omitted, the current Page will be used automatically. [optional]

This feature is new in Ignition version **8.1.3**
[Click here](#) to check out the other new features

Boolean viewportBound - If True, popups will be "shifted" to open within the bounds of the viewport. If the popup would be larger than the viewport, then it will be resized to fit within the bounds. Default is False. [optional]

- Returns

Nothing

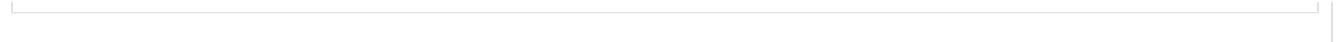
- Scope

Perspective Session

Code Examples

Code Snippet - View

```
# Toggles a popup view. We are passing in two parameters, called "myParam" and "myParam2".  
system.perspective.openPopup("myPopupId",'folder/myView', params = {"myParam":1,"myParam2":"Test"})
```



Keywords

system perspective togglePopup, perspective.togglePopup

system.perspective.vibrateDevice

This function is used in **Python Scripting**.

Description

When called from the [Perspective App](#), will cause the device to vibrate for the specified number of milliseconds.

Note: iOS vibration duration is fixed. This function will cause an iOS device to vibrate for its default duration, 0.4 seconds (400 milliseconds).

Syntax

system.perspective.vibrateDevice(integer [, sessionId])

- Parameters

String duration- The duration in milliseconds to vibrate the device.

Note: iOS vibration duration is fixed. Thus, this parameter will not impact the vibration duration on devices running iOS.

String sessionId - Identifier of the Session to target. If omitted, the current Session will be used automatically.

- Returns

Nothing

- Scope

Perspective Session

Code Examples

Code Snippet - View

```
# Vibrates the device for 1/2 second (500 milliseconds).  
system.perspective.VibrateDevice(500)
```

Keywords

system perspective vibrateDevice, perspective.vibrateDevice

system.perspective.workstation

Perspective Workstation Functions

The following functions offer various ways to interact with Perspective Workstation from a Python script. Note that the functions here only work when called from a session running in Perspective Workstation.

[In This Section ...](#)

system.perspective.workstation.exit

This function is used in **Python Scripting**.

Description

When called from a session running in Workstation, this function will close Workstation.

Syntax

system.perspective.workstation.exit()

- Parameters

Nothing

- Returns

Nothing

- Scope

Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system perspective workstation exit, perspective.workstation perspective.workstation.exit

system.perspective.workstation.toKiosk

This function is used in **Python Scripting**.

Description

When called from a session running in [Perspective Workstation](#), attempts to put Workstation into Kiosk mode.

Syntax

system.perspective.workstation.toKiosk()

- Parameters

Nothing

- Returns

Nothing

- Scope

Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system perspective workstation toKiosk perspective.workstation perspective.workstation.toKiosk

system.perspective.workstation.toWindowed

This function is used in **Python Scripting**.

Description

When called from a Session running in [Perspective Workstation](#), attempts to put Workstation into windowed mode.

Syntax

system.perspective.workstation.toWindowed()

- Parameters
 - Nothing
- Returns
 - Nothing
- Scope
 - Perspective Session

Code Examples

Code Snippet - View

```
system.perspective.workstation.toWindowed( )
```

Keywords

system perspective workstation toWindowed perspective.workstation perspective.workstation.toWindowed

system.print

Print Functions

The following functions allow you to send to a printer.

[In This Section ...](#)

system.print.createImage

This function is used in **Python Scripting**.

Description

Takes a snapshot of a component and creates a Java BufferedImage out of it. You can use [javax.imageio](#) to turn this into bytes that can be saved to a file or a BLOB field in a database.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.print.createImage(component)
```

- Parameters

[Component](#) component - The component to render.

- Returns

[BufferedImage](#) - A [java.awt.image.BufferedImage](#) representing the component.

- Scope

Vision Client

Code Examples

```
from java.io import File
from javax.imageio import ImageIO

component = event.source.parent.getComponent('TestPieChart')
bufferedImage = system.print.createImage(component)

rawPath = system.util.getProperty("user.home") + system.util.getProperty("file.separator") +
chart_createImage.jpg"
formattedPath = File(rawPath)

ImageIO.write(bufferedImage , "jpg", formattedPath)
```

Keywords

system print createImage, print.createImage

system.print.createPrintJob

This function is used in **Python Scripting**.

Description

Provides a general printing facility for printing the contents of a window or component to a printer. The general workflow for this function is that you create the print job, set the options you'd like on it, and then call `print()` on the job. For printing reports or tables, use those components' dedicated `print()` functions.

The PrintJob object that this function returns has the following properties:

Property	Description
Show Print Dialog	If true (1), then the print dialog window will be shown before printing. This allows users to specify printing options like orientation, printer, paper size, margins, etc. [default: 1]
Fit To Page	If the component is too big or small to fit on a page, it will be proportionately zoomed out or in until it fits into the page. [default: 1]
Zoom Factor	If greater than zero, this zoom factor will be used to zoom the printed image in or out. For example, if this is 0.5, the printed image will be half size. If used, this zoom factor overrides the Fit To Page parameter. [default: -1.0]
Orientation	The orientation that the page will be printing at. 1 for Portrait, 0 for Landscape. [default: 1]
Page Width	The width of the paper in inches. [default: 8.5]
Page Height	The height of the paper in inches. [default: 11]
Left Margin, Right Margin, Top Margin, Bottom Margin	The margins, specified in inches. [default: 0.75]
Printer Name	The name of the printer that this will default print to, if available.

The properties of the PrintJob object can be altered before printing the document.

Property	Retrieve the value with...	Set the value with...
Show Print Dialog	<code>.isShowPrintDialog()</code>	<code>.setShowPrintDialog(boolean)</code>
Fit To Page	<code>.isFitToPage()</code>	<code>.setFitToPage(boolean)</code>
Zoom Factor	<code>.getZoomFactor()</code>	<code>.setZoomFactor(double)</code>
Orientation	<code>.getOrientation()</code>	<code>.setOrientation(int)</code>
Page Width	<code>.getPageWidth()</code>	<code>.setPageWidth(float)</code>
Page Height	<code>.getPageHeight()</code>	<code>.setPageHeight(float)</code>
Left Margin	<code>.getLeftMargin</code>	<code>.setLeftMargin(float)</code>
Right Margin	<code>.getRightMargin()</code>	<code>.setRightMargin(float)</code>
Top Margin	<code>.getTopMargin()</code>	<code>.setTopMargin(float)</code>
Bottom Margin	<code>.getBottomMargin()</code>	<code>.setBottomMargin(float)</code>
Printer Name	<code>.getPrinterName()</code>	<code>.setPrinterName(string)</code>
All Margins*	-	<code>.setMargins(float)</code>

*All Margins isn't a property of the PrintJob, but rather all four of the PrintJob's Margins can be set at the same time using that function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.print.createPrintJob(component)

- Parameters

[Component](#) component - The component that you'd like to print. Refer to [Components](#) objects.

- Returns

[JythonPrintJob](#) - A print job that can then be customized and started. To start the print job, use .print(). Refer to [JythonPrintJob](#) object s.

- Scope

Vision Client

Code Examples

Code Snippet

```
# Put this code on a button to print out an image of the container the button is in.  
# A print dialog box will be displayed, allowing the user to specify various aspects of the print job.  
job = system.print.createPrintJob(event.source.parent)  
job.print()
```

Code Snippet

```
# Put this code on a button to print out an image of components in a container component,  
# giving very specific print options and removing the ability for the user to configure the print job.  
job = system.print.createPrintJob(event.source.parent.getComponent('Container'))  
job.setShowPrintDialog(0)  
job.setPageHeight(3)  
job.setPageWidth(5)  
job.setMargins(.5)  
job.setOrientation(0)  
job.print()
```

Keywords

system print createPrintJob, print.createPrintJob

system.print.printToImage

This function is used in [Python Scripting](#).

Description

This function prints the given component (such as a graph, container, entire window, etc) to an image file, and saves the file where ever the operating system deems appropriate. A filename and path may be provided to determine the name and location of the saved file.

While not required, it is highly recommended to pass in a filename and path. The script may fail if the function attempts to save to a directory that the client does not have access rights to.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.print.printToImage(component, [filename])

- Parameters

[Component](#) component - The component to render. Refer to the list of [Components](#) objects.

[String](#) filename - A filename to save the image as. [optional]

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would go on a button and save an image of the container that it is in.  
system.print.printToImage(event.source.parent, "C:\\temp\\Screen.jpg")
```

Code Snippet - User Selected Location

```
# Again, this example would save an image of the container, but prompts the user for a location and  
filename with system.file.saveFile()  
  
# Ask the user for a location. Uses a default filename of "image.png"  
path = system.file.saveFile("image.png")  
  
# If the path is not None...  
if path != None:  
    #Save the file  
    system.print.printToImage(event.source.parent, path)
```

Keywords

system print printToImage, print.printToImage

system.project

Project Functions

The following functions allow you to list projects on the Gateway through scripting.

[In This Section ...](#)

system.project.getProjectName

This function is used in **Python Scripting**.

Description

Returns the name of the project where the function was called from. When called from the Gateway scope from a resource that originates from a singular project (reports, SFCs, etc.), will return that resources project. Resources that run in the Gateway scope, but are configured in a singular project (such as a report), will use that project's name.

When called from a scope that does not have an associated project (a Tag Event Script), the function will return the name of the Gateway scripting project. If a Gateway scripting project has not been configured, then returns an empty string.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.project.getProjectName()
```

- Parameters
 - Nothing
- Returns
 - String** - The name of the currently running project.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code displays the name of the currently running project to the appropriate console, depending on
# scope (Designer console, Gateway console, etc.).
system.util.getLogger("myLogger").warn("You are running project: %s" % system.project.getProjectName())
```

Keywords

system project getprojectname, project.getprojectname

system.project.getProjectNames

This function is used in **Python Scripting**.

Description

Returns an unsorted collection of strings, where each string represents the name of a project on the Gateway. If no projects exist, returns an empty list.

This function only ever returns project names, ignoring project titles. The function also ignores the "enabled" property, including disabled projects in the results.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.project.getProjectNames()
```

- Parameters

- Nothing

- Returns

- [List](#) - A list containing string representations of project names on the Gateway.

- Scope

- Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Calling this from the Script Console prints out each project name.  
print system.project.getProjectNames()
```

Keywords

system project getprojectnames, project.getprojectnames

system.report

Report Functions

The following functions give you access to report details and the ability to run reports.

[In This Section ...](#)

system.report.executeAndDistribute

This function is used in [Python Scripting](#).

Description

Executes and distributes a report. Similar to [scheduling a report to execute](#), except a schedule is not required to utilize this function. This is a great way to distribute the report on demand from a Client. Throws an `IllegalArgumentException` when any of the following occurs: If the file type is not recognized, path does not exist, project does not exist, or a key is not valid.

Note: The function `system.report.executeAndDistribute()` does not run on its own thread and can get blocked. For example, if a printer is backed up and it takes a while to finish the request made by this function, the script will block the execution of other things on that thread until it finishes. Be sure to keep this in mind when using it in a script.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

 This function accepts [keyword arguments](#).

system.report.executeAndDistribute(path, project, [parameters], action, [actionSettings])

- Parameters

`String path` - The path to the existing report.

`String project` - The name of the project where the report is located. Optional in Client scope. Optional in Session scope.

`Dictionary[String, Integer] parameters` - A dictionary of parameter overrides, in the form name:value pairs. [optional]

`String action` - The name of the distribution action to use. The action parameter supports the following keys as strings:

- email
- print
- save
- ftp

`Dictionary[List, Any] actionSettings` - A dictionary of settings particular to the action. Missing values will use the default value for that action. [optional]

- Returns

`Nothing`

- Scope

Gateway, Vision Client, Perspective Session

Values for actionSettings

The action settings parameter supports an optional dictionary of settings particular to the action. Missing values will use the default value for that action.

Note: The email action now has the ability to add emails to the reply to field of the email. The `replyTo`, `replyToRoles`, and `replyToUserSource` keys have been added to the possible dictionary options.

- email
 - Setting Keys: "smtpServerName", "from", "subject", "body", "attachmentName", "retries", "fileType", "to", "cc", "bcc", "replyTo", "useRoles", "roles", "userSource", "replyToRoles", "replyToUserSource".

- Note: *To*, *cc*, *bcc*, and *replyTo* must be Python lists. If *useRoles* is True, *to*, *cc* and *bcc* will be ignored and all email addresses for all users matching *roles* in *userSource* (which defaults to the project's current user source) will be in the *to* field. Similarly, all users matching the *replyToRoles* in *replyToUserSource* will be in the *reply to* field of the email. If *useRoles* is true but no *roles* are listed, all user email addresses in *userSource* will be in the *to* field. If omitted, *fileType* defaults to pdf.
- print
 - Setting Keys: "primaryPrinterName", "backupPrinterName", "copies", "printBothSides", "collate", "useRaster", "rasterDPI", "useAutoLandscape", "pageOrientation".
 - Note: *primaryPrinterName* defaults to the default printer. *backupPrinterName* defaults to "none", but can also have the special value of "default". *printBothSides*, *collate*, and *useRaster* are booleans which default to false. *rasterDPI* is only used if *useRaster* is true. *useAutoLandscape* defaults to true. If *useAutoLandscape* is false, *pageOrientation*, which can have values of "portrait" or "landscape" (default is "portrait"), is used.
- save
 - Setting Keys: "path", "fileName" and "format".
 - Note: Since the script is sent to the Gateway for execution, path and fileName must be relative to the Gateway.
- ftp
 - Setting Keys: "server", "port", "username", "password", "useSSL", "path", "fileName", and "format".
 - Note: Server and fileName are required. If omitted, fileType defaults to pdf, port defaults to 21, and useSSL defaults to false.

Values for the fileType Parameter

Acceptable values are:

- pdf (recommended)
- html
- csv
- rtf
- jpeg
- png
- xml
- xls
- xlsx

A Note on xls and xlsx Formats

The xls and xlsx format options may return less than pixel perfect results. This is due to how many spreadsheet programs interpret the resulting file. As a result, the pdf format is recommended in most cases.

Code Examples

Code Snippet - Emailing a Report

```
# Executes and distributes the report to an email address.
system.report.executeAndDistribute(path="My Report Path", project="My Project", action= "email",
    actionSettings = {"to":["plantmanager@myplant.com"], "smtpServerName":"myplantMailServer", "from": "reporting@myplant.com", "subject": "Production Report"})
```

Code Snippet - Emailing a Report

```
# Executes and distributes the report to all users in the default user source who are Supervisors or Managers.
system.report.executeAndDistribute(path="My Report Path", project="My Project", action= "email",
    actionSettings = {"useRoles":True, "roles":["Supervisor", "Manager"], "smtpServerName": "myplantMailServer", "from": "reporting@myplant.com", "subject": "Production Report"})
```

Code Snippet - Sending Report to FTP Server

```
# Executes and distributes the report to an ftp server with parameter values passed into the report
reportParameters = {"StartDate":system.date.addHours(system.date.now(), -12), "EndDate":system.date.now()}
settings = {"server": "10.20.1.80", "port": 22, "username": "Ignition", "password": "Secret", "useSSL": False, "path": "C:\\\\FTP", "fileName": "Ignition Report", "format": "pdf"}
system.report.executeAndDistribute(path="My Report Path", project="My Project",
parameters=reportParameters, action= "ftp", actionSettings = settings)
```

Code Snippet - Saving Report

```
# Executes and distributes the report to save a PDF
settings = { "path": "C:\\Ignition Reports", "fileName": "Report.pdf", "format": "pdf" }
system.report.executeAndDistribute(path="My Report Path", project="My Project", action="save",
actionSettings=settings)
```

Keywords

system report executeAndDistribute, report.executeAndDistribute

system.report.executeReport

This function is used in [Python Scripting](#).

Description

Immediately executes an existing report and returns a List[Byte] of the output. Throws an ArgumentException when any of the following occurs: If the file type is not recognized, path does not exist, project does not exist.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

 This function accepts [keyword arguments](#).

system.report.executeReport(path, project, [parameters], fileType)

- Parameters

[String](#) path - The path to the existing report.

[String](#) project - The name of the project where the report is located. Optional in Vision Client scope and Perspective Session scope.

[Dictionary\[String, Integer\]](#) parameters - A dictionary of parameter overrides, in the form name:value. [optional]

[String](#) fileType - The file type the resulting byte array should represent. Defaults to "pdf". Not case-sensitive.

- Returns

[List\[Byte\]](#) - A byte array of the resulting report.

- Scope

Gateway, Vision Client, Perspective Session

Values for the fileType Parameter

Acceptable values are:

- pdf
- html
- csv
- rtf
- jpeg
- png
- xml
- xls
- xlsx

A Note on xls and xlsx Formats

The xls and xlsx format options may return less than pixel perfect results. This is due to how many spreadsheet programs interpret the resulting file. As a result, the pdf format is recommended in most cases.

Code Examples

Code Snippet - Executing Report

```
# Executes the report, overriding two parameters.  
overrides = {"myStringParam": "Hello world", "myIntParam": 3}  
bytesArray = system.report.executeReport(path="My Path", project="MyProject", parameters=overrides,  
fileType="pdf")
```

Keywords

system report executeReport, report.executeReport

system.report.getReportNamesAsDataset

This function is used in **Python Scripting**.

Description

Gets a data of all reports for a project. Throws an `IllegalArgumentException` when any of the following occurs: If the project name is omitted in the Gateway scope, project does not exist.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

 This function accepts [keyword arguments](#).

system.report.getReportNamesAsDataset (project)

- Parameters

`String project` - The name of the project where the reports are located. Optional in Client scope. Optional in Session scope.

- Returns

`Dataset` - A dataset of report paths and names for the project. Return columns are Path, Text, and SelectedText. Returns an empty dataset if the project has no reports.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Gets a dataset of reports for the current project and displays
# them in a Tree View component.

event.source.parent.getComponent('Tree View').data = system.report.getReportNamesAsDataset()
```

Keywords

system report getReportNamesAsDataset, report.getReportNamesAsDataset

system.report.getReportNamesAsList

This function is used in [Python Scripting](#).

Description

Gets a list of all reports for a project. Throws an `IllegalArgumentException` when any of the following occurs: If the project name is omitted in the Gateway scope or if project does not exist.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

 This function accepts [keyword arguments](#).

system.report.getReportNamesAsList(project)

- Parameters

[String](#) `project` - The name of the project where the reports are located. Optional in Client scope and Perspective Session scope.

- Returns

[List](#) - A list of report paths for the project. Returns an empty list if the project has no reports.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Gets a list of reports for the current project and prints it.
reports = system.report.getReportNamesAsList()
for report in reports:
    print report

"""Output from the above example looks like the following:
Comparisons
Line Reports/Line 1/Defect rates
Line Reports/Line 1/Production
Line Reports/Line 2/Defect Rates
"""


```

Keywords

`system report getReportNamesAsList, report.getReportNamesAsList`

system.roster

Roster Functions

Functions that provide roster manipulation, including adding and remove users from a roster.

[In This Section ...](#)

system.roster.addUsers

This function is used in **Python Scripting**.

Description

Adds a list of users to an existing roster. Users are always appended to the end of the roster.

Syntax

system.roster.addUsers(rosterName, [users])

- Parameters

String rosterName- The name of the roster to modify.

List users - A list of [User](#) objects that will be added to the end of the roster. User objects can be created with the [system.user.getUser](#) and [system.user.addUser](#) functions. These users must exist before being added to the roster.

- Returns

Nothing

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
# Adds a couple of users to a roster.  
userSource = "default"  
rosterName = "rosterEast"  
  
# getUser() returns a user object, which is needed for addUser()  
userA = system.user.getUser(userSource, "george")  
userB = system.user.getUser(userSource, "joe")  
  
system.roster.addUsers(rosterName, [userA, userB])
```

Keywords

system roster addUsers, roster.addUsers

system.roster.createRoster

This function is used in **Python Scripting**.

Description

Creates a roster with the given name and description, if it does not already exist.

This function was designed to run in the Gateway and in Perspective sessions. If creating rosters from Vision clients, use [system.alarm.createRoster](#) instead

Syntax

system.roster.createRoster(name, description)

- Parameters

String name - The name of the roster to create.

String description - The description for the roster. May be None, but the parameter is mandatory.

- Returns

Nothing

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
# Create an empty roster with a description.  
system.roster.createRoster("rosterEast", "East plant user roster")
```

```
# Create an empty roster roster without a description.  
system.roster.createRoster("rosterWest", None)
```

Keywords

system roster createRoster, roster.createRoster

system.roster.deleteRoster

This function is used in **Python Scripting**.

This feature is new in Ignition version **8.1.1**
[Click here](#) to check out the other new features

Description

Deletes a roster with the given name.

Syntax

system.roster.deleteRoster(rosterName)

- Parameters

String name - The name of the roster to delete.

- Returns

Nothing

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
system.roster.deleteRoster("some roster")
```

Keywords

system roster deleteRoster, roster.deleteRoster

system.roster.getRosters

This function is used in **Python Scripting**.

Description

Returns a dictionary of rosters, where the key is the name of the roster, and the value is an array list of string user names.

This function was designed to run in the Gateway and in Perspective sessions. If creating rosters from Vision clients, use [system.alarm.getRosters](#) instead.

Syntax

system.roster.getRosters()

- Parameters

Nothing

- Returns

Dictionary[String, List[String]] - A dictionary that maps roster names to a list of usernames in the roster. The list of usernames may be empty if no users have been added to the roster.

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
# This example prints out all existing rosters to the console of a Perspective session:  
  
rostros = system.roster.getRosters()  
  
# Iterate over the rosters, extracting the name and user lists.  
for name, users in rostros.items():  
  
    # Format the results in a somewhat presentable manner.  
    msg = "{0} : {1}".format(name, users)  
  
    # Output the result.  
    system.perspective.print(msg)
```

Get Each User in a Roster

```
# This example prints out each user in a certain roster.  
rostros = system.roster.getRosters()  
  
# Specify the roster with the key (the name of the roster), and iterate over the users.  
for user in rostros['myRoster']:   
  
    # Output the users.  
    system.perspective.print(user)
```

Keywords

system roster getRosters, roster.getRosters

system.roster.removeUsers

This function is used in **Python Scripting**.

Description

Removes one or more users from an existing roster.

Syntax

system.roster.removeUsers(rosterName, users)

- Parameters

String rosterName- The name of the roster to modify.

List users - A list of user objects that will be removed from the roster. User objects can be created with the [system.user.getUser](#) and [system.user.addUser](#) functions.

- Returns

Nothing

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
userSource = "default"
rosterName = "rosterEast"

# getUser() returns a user object, which is needed for removeUser()
user = system.user.getUser(userSource, "joe")

system.roster.removeUsers(rosterName, [user])
```

Keywords

system roster removeUsers, roster.removeUsers

system.secsgem

SECS/GEM Functions

The following functions allow you to interact with equipment defined by the SECS/GEM module. Note that the [SECS/GEM](#) module must be installed before these functions will be accessible.

[In This Section ...](#)

system.secsgem.copyEquipment

This function is used in **Python Scripting**.

Description

Creates a copy of an equipment connection. Common settings can be overridden for the new connection. An exception is thrown if the new equipment connection cannot be created.

Client Permission Restrictions

Permission Type: SECS/GEM Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when called in the Gateway scope.

Syntax



This function accepts **keyword arguments**.

system.secsgem.copyEquipment(equipmentSource, newEquipmentName, enabled, activeAddress, activePort, passiveAddress, passivePort, deviceld, [dbTablePrefix], [description])

- Parameters

String equipmentSource - Some new equipment settings will be retrieved from this equipment connection. Specify the source equipment connection name.

String newEquipmentName - The name of the new equipment connection.

Boolean enabled - If set to false, the new equipment connection will be disabled after it is created.

String activeAddress - IP Address of new equipment. Must be specified if the SECS/GEM module is used in Active mode. Otherwise, do not use this parameter.

Integer activePort - Port number of new equipment. Must be specified if the SECS/GEM module is used in Active mode. Otherwise, do not use this parameter.

String passiveAddress - IP Address of new equipment. Must be specified if the SECS/GEM module is used in Passive mode. Otherwise, do not use this parameter.

Integer passivePort - Port number of new equipment. Must be specified if the SECS/GEM module is used in Passive mode. Otherwise, do not use this parameter.

Integer deviceld - Unique identifier of new equipment. This value must be an integer, and is specified within the equipment.

String dbTablePrefix - SECS/GEM database table names will use the specified prefix for the new equipment connection. If no prefix is specified, the description of the source equipment will be used. [optional]

String description - The description for the new equipment connection. If no description is specified, the description of the source equipment will be used. [optional]

- Returns

Nothing

- Scope

Designer, Vision Client

The Address and Port Parameters

When calling this function, only one set of address and port parameters need to be specified: Either activeAddress and activePort, or passiveAddress and passivePort.

Optionally, both sets of parameters may be specified, but the function will throw an exception if neither are specified.

Code Examples

Code Snippet - Copy Equipment

```
system.secsgem.copyEquipment(equipmentSource="ToolOne", newEquipmentName="ToolTwo", enabled=True,  
activeAddress="192.168.1.5", activePort=15500, deviceId=0)
```

Keywords

system secsgem copyEquipment, secsgem.copyEquipment

system.secsgem.deleteToolProgram

This function is used in **Python Scripting**.

Description

Deletes a process program from the Gateway.

Client Permission Restrictions

Permission Type: SECS/GEM Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when called in the Gateway scope.

Syntax

system.secsgem.deleteToolProgram(ppid)

- Parameters
 - String ppid - The PPID that was sent from the tool when the S7F3 message was saved.
- Returns
 - Nothing
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Name of the Tool Program that will be deleted.  
targetProgram = "Old Program"  
  
system.secsgem.deleteToolProgram(targetProgram)
```

Keywords

system secsgem deleteToolProgram, secsgem.deleteToolProgram

system.secsgem.enableDisableEquipment

This function is used in **Python Scripting**.

Description

Enables or disables a Tuple of equipment connections from a script.

Client Permission Restrictions

Permission Type: SECS/GEM Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when called in the Gateway scope.

Syntax

system.secsgem.enableDisableEquipment(enable, names)

- Parameters

Boolean enable - Set to True to enable equipment connections, or set to False to disable them.

Tuple names - A Tuple of strings. Each string should match an equipment connection configured on the Gateway. If this parameter contains the name of an equipment connection that does not exist, then a message will be included in the List returned by this function.

- Returns

List - A list of unicode strings. Each string contains a message about an equipment connection that could not be enabled/disabled. If the list is empty, then all specified equipment connections were successfully modified.

- Scope

Designer, Vision Client

Code Examples

Code Snippet - Disabling Equipment

```
# Executing this example script will attempt to disable two equipment connections.

# Create a Python Tuple of equipment names to disable.
equipmentNames = ("ToolOne", "ToolTwo")

# Invoke the Function.
result = system.secsgem.enableDisableEquipment(False, equipmentNames)

# Print the results of any equipment connections that could not be modified.
print result
```

Keywords

system secsgem enableDisableEquipment, secsgem.enableDisableEquipment

system.secsgem.getResponse

This function is used in [Python Scripting](#).

Description

Attempts to retrieve a response message from the Gateway. The transaction id from the sent message is used to retrieve the response.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.getResponse(transactionID, equipment, [timeout], [poll])

- Parameters

Integer transactionID - The transactionID of the request and response. The transactionID is used to retrieve the response. Typically used in conjunction with [system.secsgem.sendRequest](#) to generate a transactionID.

String equipment - Name of equipment connection.

Integer timeout - Specifies in seconds how long to wait for a response before returning None. If omitted the timeout will be 5 seconds. [optional]

Integer poll - Specifies in milliseconds how often to poll the system for a response. If omitted the poll will be 150 milliseconds. [optional]

- Returns

Any - A Python object, typically a dictionary. The actual result is a JSON string that's decoded into a Python object, as shown on the mapping on the [system.util.jsonDecode](#) page.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Replace the string below with the equipment name you want to send the request to.  
myEquipment = "EquipmentOne"  
  
# Define the contents of the body. We're using an empty string, since S1F1 doesn't expect a body, and we  
need to define something (Python's None will result in an exception).  
body = ""  
  
# Store the returned transactionID in a variable.  
transactionID = system.secsgem.sendRequest("S1F1", True, body, myEquipment)  
  
# Use the transactionID to lookup the response.  
response = system.secsgem.getResponse(transactionID, myEquipment, 2)  
  
# We're printing out the response here, but you could do something more useful instead.  
print response
```

Code Snippet

```
# This example demonstrates how to retrieve the value of a Status Variable via S1F3.  
# If using the simulator that comes with the SECS/GEM module, this example will return the current time
```

```

from the Clock Status Variable.

# Replace the string below with the equipment name you want to send the request to.
myEquipment = "EquipmentOne"

# Define the contents of the body. The Clock Status Variable has an SVID of 1.
body = [{"format": "U4", "value": 1}]

# Store the returned transactionID in a variable.
transactionID = system.secsgem.sendRequest("S1F3", True, body, myEquipment)

# Retrieve the response.
response = system.secsgem.getResponse(transactionID, myEquipment, 2)

## We need to do some digging to get the value of the Clock:
## -The response is a Dictionary.
## -Inside of the response is the key "body".
## -The value of "body" is a Python List containing another Dictionary (which has our Clock value)
##           Thus we use [0] to access the Dictionary.
## -The Dictionary contains a key named "value", which is the value of our clock.
theDatetime = response["body"][0]["value"]

# We parse the date into something more human readable, and print it out.
print system.date.parse(theDatetime, "yyMddHHmmss")

```

Keywords

system secsgem getResponse, secsgem.getResponse

system.secsgem.getToolProgram

This function is used in [Python Scripting](#).

Description

Returns a process program from the Gateway that was previously sent by a tool in an S7F3 message.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.getToolProgram(ppid)

- Parameters

[String ppid](#) - The PPID that was sent from the tool when the S7F3 message was saved.

- Returns

[Dictionary](#) - A Python Dictionary containing the following keys: [editDate, ppbody, bodyFormat].

- **'editDate'** - Holds the last date the program was saved.
- **'ppbody'** - Holds the actual program.
- **'bodyFormat'** - Holds the format ('A', 'B', 'I', etc) of the original message PPBODY.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Getting Tool Program

```
# Retrieve information on all programs, and convert them to a PyDataset.  
# PyDatasets are easier to iterate over.  
results = system.secsgem.getToolProgramDataset()  
pyResults = system.dataset.toPyDataSet(results)  
  
for program in pyResults:  
    # If the format of the program is ASCII...  
    if program[2] == "A":  
  
        ppid = program[0]  
        # ...retrieve more information on the program...  
        programData = system.secsgem.getToolProgram(ppid)  
        # ...and print the program. Writing to a file would most  
        # likely be a better practice here.  
        print "Program %s: %s" % (ppid,programData[1])
```

Keywords

system secsgem getToolProgram, secsgem.getToolProgram

system.secsgem.getToolProgramDataset

This function is used in [Python Scripting](#).

Description

Returns a dataset containing information about all stored process programs.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.getToolProgramDataset()

- Parameters

Nothing

- Returns

Dataset - A dataset containing information about all stored process programs. Includes the following columns in order: **ppid**, **editDate**, **bodyFormat**.

- **ppid** - The name (PPID) of the program.
- **editDate** - The last known date the program was saved.
- **bodyFormat** - The format of the program. Uses notation matching SECS item definitions: "A" for ASCII, "B" for binary, etc.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Retrieve information about all programs.  
results = system.secsgem.getToolProgramDataset()  
  
# Convert the dataset to a PyDataset, since they are easier to iterate over.  
pyResults = system.dataset.toPyDataSet(results)  
for program in pyResults:  
  
    # Print out details on each program.  
    print "Program %s was last modified on %s" % (program[0], program[1])
```

Keywords

system secsgem getToolProgramDataset, secsgem.getToolProgramDataset

system.secsgem.sendRequest

This function is used in [Python Scripting](#).

Description

Sends a JSON-formatted SECS message to a tool. An equipment connection must be configured for the tool in the Gateway.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.sendRequest(streamFunction, reply, body, equipment)

- Parameters

[String](#) streamFunction - The stream and function of the SECS message to send, for example: "S1F13".

[Boolean](#) reply - Whether or not the SECS message expects a reply message.

[Any](#) body - This contains the body of a SECS message. The argument can be a Python Object or JSON string representing the body of a SECS message. If this argument is a string then it will be converted to a Python Object using the [system.util.jsonDecode](#) function.

[String](#) equipment - Name of the equipment connection to use.

- Returns

[Integer](#) - The transactionID of the SECS message response.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Sending a S1F1 Message

```
# Replace the string below with the equipment name you want to send the request to.  
myEquipment = "EquipmentOne"  
  
# Define the contents of the body. We're using an empty string, since S1F1 doesn't expect a body, and we  
need to define something (Python's None will result in an exception).  
body = ""  
  
# Store the returned transactionID in a variable. This script could be extended by using system.secsgem.  
getResponse to view the response.  
transactionID = system.secsgem.sendRequest("S1F1", True, body, myEquipment)
```

Keywords

system secsgem sendRequest, secsgem.sendRequest

system.secsgem.startSimEventRun

This function is used in **Python Scripting**.

Description

Starts a configured simulator event run in the Gateway. Note, that this function only works with the simulators that come included with the SECS /GEM module.

The function will throw an exception if the specified Event Run cannot be started.

Client Permission Restrictions

Permission Type: SECS/GEM Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when called in the Gateway scope.

Syntax

system.secsgem.startSimEventRun(simulatorName, eventRunName)

- Parameters

String simulatorName - The simulator that holds the configured event run. Will throw an exception if the specified simulator can't be found.

String eventRunName - The event run to start. Will throw an exception if the specified simulator can't be found.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This examples requires that the Gateway has a simulator named "simulator1", and
# an Event Run in that same simulator named "myEventRun".
mySimulator = "simulator1"
eventRun = "myEventRun"

system.secsgem.startSimEventRun(mySimulator, eventRun)
```

Keywords

system secsgem startSimEventRun, secsgem.startSimEventRun

system.secsgem.toDataSet

This function is used in [Python Scripting](#).

Description

Converts a SECS message data structure, as returned by the [system.secsgem.getResponse](#) function, into a dataset and returns it.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.toDataSet(secsObject)

- Parameters

[Any](#) secsObject - A Python object, such as sequence or a dictionary, representing a SECS message to convert to a dataset. More information on how to format the Python object can be found on the [SECS Definition Language \(SDL\) File](#) page.

- Returns

[Dataset](#) - A dataset representing a SECS message.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Receive a SECS message. Example assumes you have ha valid transaction ID.  
object = system.secsgem.getResponse(transactionID, "myEquipment", 2)  
  
# Turn the message into a dataset.  
dataset = system.secsgem.toDataSet(object)  
  
# Assuming this script was called from a component script, and a Table component was in the same  
# container as the component that called this script, we could pass  
# the dataset to the Data property.  
event.source.parent.getComponent('Table').data = dataset
```

Code Snippet - Manually Making the Message

```
{  
    "header": {  
        "doc": "nonexistent function",  
        "stream": 100,  
        "function": 100,  
        "reply": "False"  
    },  
    "body": [  
        {  
            "doc": "FirstItem, my first nonsense item",  
            "format": "U4",  
            "value": 1234  
        },  
        {  
            "doc": "SecondItem, the other nonsense item",  
            "format": "U4",  
            "value": 5678  
        }  
    ]  
}
```

```
        "format": "U4",
        "value": 5678
    }
]
```

Keywords

system secsgem toDataSet, secsgem.toDataSet

system.secsgem.toTreeDataSet

This function is used in [Python Scripting](#).

Description

Changes an existing dataset, as returned by the [system.secsgem.toDataSet](#) function, to make it usable for the [Vision Tree View](#) component.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.toTreeDataSet(dataset)

- Parameters

Dataset dataset - A dataset containing a SECS message. Note that this parameter cannot take a JSON message, so the object returned by [system.secsgem.getResponse](#) must first be processed by [system.secsgem.toDataSet](#).

- Returns

Dataset - A dataset containing a SECS message with the following columns, as suited for Vision's tree view component: "path", "text", "icon", "background", "foreground", "tooltip", "border", "selectedText", "selectedIcon", "selectedBackground", "selectedForeground", "selectedTooltip", "selectedBorder"

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Assuming the variable named "message" contains a SECS message, we will first convert it to a Dataset.  
dataset = system.secsgem.toDataSet(message)  
  
# The initial dataset generated by toDataSet will not work with the Tree View component, so we'll modify  
it...  
dataset = system.secsgem.toTreeDataSet(dataset)  
  
# ...and now pass the dataset into the Tree View component's data property.  
event.source.parent.getComponent('Tree View').data = dataset
```

Keywords

`system secsgem toTreeDataSet, secsgem.toTreeDataSet`

system.secsgem.sendResponse

This function is used in [Python Scripting](#).

Description

Sends a JSON-formatted SECS response message to a message sent by a tool. An equipment connection must be configured for the tool in the Gateway, and this must be used within a Message Handler to create a [Custom Message Response Handler](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.secsgem.sendResponse(transactionID, systemBytes, streamFunction, body, equipment)

- Parameters

[Integer](#) transactionID - The TxID of the response. The TxID from the received request in the payload of the message handler must be specified here.

[Integer](#) systemBytes - The SystemBytes of the response. The SystemBytes from the received request in the payload of the message handler must be specified here.

[String](#) streamFunction - The stream and function of the SECS message to send, for example: "S1F14".

[Any](#) body - This contains the body of a SECS response. The argument can be a Python object or JSON string representing the body of a SECS message. If this argument is a string then it will be converted to a Python Object using the [system.util.jsonDecode](#) function.

[String](#) equipment - Name of the equipment connection to use.

- Returns

Nothing

- Scope

Gateway

Code Examples

Code Snippet - Sending a S1F1 Message

```
# This will create a logger that will print to the console that a custom response is happening for S6F12.
# It will then send the response with system.secsgem.sendResponse().
equipment= payload['Equipment']
txId = payload['TxID']
systemBytes = payload['SystemBytes']
message = payload['Message']

msg = "Equipment=" + equipment
msg += ", TxID=" + str(txId)
msg += ", SystemBytes=" + str(systemBytes)
msg += ", Message=" + message
logger = system.util.getLogger("SECSGEM.Gateway.S6F12Handler")
logger.info("S6F12Handler: Sending back response to S6F11 message:" + msg)

body = '{"format":"B", "value": 0, "doc":"ACKC6, Acknowledge Code", "codeDesc": "Accepted"}'
system.secsgem.sendResponse(txId, systemBytes, "S6F12", body, equipment)
logger.info("S6F12Handler: S612 response sent")
```

Keywords

system secsgem sendResponse, secsgem.sendResponse

system.security

Security Functions

The following functions give you access to interact with the users and roles in the Gateway. These functions require the Vision module, as these functions can only be used with User Sources and their interaction with Vision Clients.

[In This Section ...](#)

system.security.getRoles

This function is used in [Python Scripting](#).

Description

Finds the roles that the currently logged in user has, returns them as a Python tuple of strings.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.getRoles()

- Parameters
 - Nothing
- Returns
 - [Tuple](#) - A list of the roles (strings) that are assigned to the current user.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This runs on a button to prevent certain users from opening a window.

if "Supervisor" in system.security.getRoles():
    system.nav.openWindow("ManagementOnly")
else:
    system.gui.errorBox("You don't have sufficient privileges to continue")
```

Keywords

system security getRoles, security.getRoles

system.security.getUsername

This function is used in [Python Scripting](#).

Description

Returns the currently logged-in username.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.getUsername()

- Parameters
 - Nothing
- Returns
 - [String](#) - The current user name.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code runs on a startup script and do special logic based upon who was logging in.  
name = system.security.getUsername()  
if name == 'Bob':  
    system.nav.openWindow( "BobsHomepage" )  
else:  
    system.nav.openWindow( "NormalHomepage" )
```

Keywords

system security getUsername, security.getUsername

system.security.getUserRoles

This function is used in [Python Scripting](#).

Description

Fetches the roles for a user from the Gateway. This may not be the currently logged in user. Requires the password for that user. If the authentication profile name is omitted, then the current project's default authentication profile is used.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.getUserRoles(username, password, [authProfile], [timeout])

- Parameters

[String](#) username - The username to fetch roles for.

[String](#) password - The password for the user.

[String](#) authProfile - The name of the authentication profile to run against. Leaving this out will use the project's default profile. [optional]

[Integer](#) timeout - Timeout for Client-to-Gateway communication. Default: 60,000ms. [optional]

- Returns

[Tuple](#) - A list of the roles that this user has, if the user authenticates successfully. Otherwise, returns None.

- Scope

Vision Client

Syntax

system.security.getUserRoles(username, password, [authProfile])

- Parameters

[String](#) username - The username to fetch roles for.

[String](#) password - The password for the user.

[String](#) authProfile - The name of the authentication profile to run against. Leaving this out will use the project's default profile. [optional]

- Returns

[Tuple](#) - A list of the roles that this user has, if the user authenticates successfully. Otherwise, returns None.

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
# Fetch the roles for a given user, and check to see if the role "Admin" is in them.
```

```
reqRole = "Admin"  
username = "Billy"  
password= "Secret"
```

```
roles = system.security.getUserRoles(username, password)
if reqRole in roles:
    # do something requiring "Admin" role.
```

Keywords

system security getuserRoles, security.getuserRoles

system.security.isScreenLocked

This function is used in [Python Scripting](#).

Description

Returns whether or not the screen is currently locked.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.isScreenLocked()

- Parameters
 - Nothing
- Returns
 - [Boolean](#) - A flag indicating whether or not the screen is currently locked.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This would run in a timer script to lock the screen after 15 seconds of inactivity, and then log the user out after 30 seconds of inactivity.

if system.util.getInactivitySeconds() > 15 and not system.security.isScreenLocked():
    system.security.lockScreen()
elif system.util.getInactivitySeconds() > 30:
    system.security.logout()
```

Keywords

system security isScreenLocked, security.isScreenLocked

system.security.lockScreen

This function is used in [Python Scripting](#).

Description

Used to put a running Client in lock-screen mode. The screen can be unlocked by the user with the proper credentials, or by scripting via the [system.security.unlockScreen](#) function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.lockScreen([obscure])

- Parameters

[Boolean](#) obscure - If true, the locked screen will be opaque, otherwise it will be partially visible. [optional]

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# This would run in a timer script to lock the screen after 15 seconds of inactivity, and then log the user out after 30 seconds of inactivity.

if system.util.getInactivitySeconds() > 15 and not system.security.isScreenLocked():
    system.security.lockScreen()
elif system.util.getInactivitySeconds() > 30:
    system.security.logout()
```

Keywords

system security lockScreen, security.lockScreen

system.security.logout

This function is used in [Python Scripting](#).

Description

Logs out of the Client for the current user and brings the Client to the login screen.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.logout()

- Parameters
 - Nothing
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This would run in a timer script to log the user out after 30 seconds of inactivity.  
if system.util.getInactivitySeconds() > 30:  
    system.security.logout()
```

Keywords

system security logout, security.logout

system.security.switchUser

This function is used in [Python Scripting](#).

Description

Attempts to switch the current user on the fly. If the given username and password fail, this function will return false. If it succeeds, then all currently opened windows are closed, the user is switched, and windows are then re-opened in the states that they were in.

If an event object is passed to this function, the parent window of the event object will not be re-opened after a successful user switch. This is to support the common case of having a switch-user screen that you want to disappear after the switch takes place.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.switchUser(username, password, event, [hideError])

- Parameters

[String](#) username - The username to try and switch to.

[String](#) password - The password to authenticate with.

[EventObject](#) event - If specified, the enclosing window for this event's component will be closed in the switch user process. Refer to the list of [Event](#) objects.

[Boolean](#) hideError - If True, no error will be shown if the switch user function fails. Default is False. [optional]

- Returns

[Boolean](#) - False if the switch user operation failed, True otherwise.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This script would go on a button in a popup window used to switch users without logging out of the Client.

# Pull the username and password from the input components.
uname = event.source.parent.getComponent("Username").text
pwd = event.source.parent.getComponent("Password").text

# Call switchUser. The event object is passed to this
# function so that if the username and password work,
# this window will be closed before the switch occurs.
success= system.security.switchUser(uname,pwd,event)

# If the login didn't work, give input focus back to the
# username component, so that the user can try again.
if not success:
    event.source.parent.getComponent("Username").requestFocusInWindow()
```

Keywords

system security switchUser, security.switchUser

system.security.unlockScreen

This function is used in [Python Scripting](#).

Description

Unlocks the client, if it is currently in lock-screen mode.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.unlockScreen()

- Parameters

Nothing

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would go in a global script to automatically unlock the screen on a specific computer.  
  
comp = system.net.getHostName()  
if comp == 'Line 1':  
    system.security.unlockScreen()
```

Keywords

system security unlockScreen, security.unlockScreen

system.security.validateUser

This function is used in [Python Scripting](#).

Description

Tests credentials (username and password) against an authentication profile. Returns a boolean based upon whether or not the authentication profile accepts the credentials. If the authentication profile name is omitted, then the current project's default authentication profile is used.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.security.validateUser(username, password, [authProfile], [timeout])

- Parameters

[String](#) username - The username to validate.

[String](#) password - The password for the user.

[String](#) authProfile - The name of the authentication profile to run against. Leaving this out will use the project's default profile. [optional]

[Integer](#) timeout - Timeout for Client-to-Gateway communication. Default is 60,000ms). [optional]

- Returns

[Boolean](#) - False if the user failed to authenticate; True if the username/password was a valid combination.

- Scope

Vision Client

Syntax

system.security.validateUser(username, password, [authProfile])

- Parameters

[String](#) username - The username to validate.

[String](#) password - The password for the user.

[String](#) authProfile - The name of the authentication profile to run against. Optional. Leaving this out will use the project's default profile. [optional]

- Returns

[Boolean](#) - False if the user failed to authenticate; True if the username/password was a valid combination.

- Scope

Gateway, Perspective Session

Code Examples

Code Snippet

```
# This would require the current user to enter their password again before proceeding.

currentUser = system.security.getUsername()
password = system.gui.passwordBox("Confirm Password")
valid = system.security.validateUser(currentUser, password)
```

```
if valid:  
    # Do something.  
else:  
    system.gui.errorBox("Incorrect password")
```

Keywords

system security validateUser, security.validateUser

system.serial

Serial Functions

The following functions give you access to read and write through serial ports.

[In This Section ...](#)

system.serial.closeSerialPort

This function is used in **Python Scripting**.

Description

Closes a previously opened serial port. Returns without doing anything if the named serial port is not currently open. Will throw an exception if the port is open and cannot be closed.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.closeSerialPort(port)

- Parameters

String port - The name of the serial port, e.g., "COM1" or "dev/ttyS0".

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system serial closeSerialPort, serial.closeSerialPort

system.serial.configureSerialPort

This function is used in **Python Scripting**.

Description

Configure a serial port for use in a later call. This only needs to be done once unless the configuration has changed after the initial call. All access to constants must be prefixed by "system.serial.".

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

 This function accepts [keyword arguments](#).

system.serial.configureSerialPort(port, [bitRate], [dataBits], [handshake], [hardwareFlowControl], [parity], [stopBits])

- Parameters

String port - The name of the serial port (e.g., "COM1" or "/dev/ttyS0"). This parameter is required.

Integer bitRate - Configure the bit rate. Valid values are defined by the following constants [optional]:

```
system.serial.BIT_RATE_110, system.serial.BIT_RATE_150, system.serial.BIT_RATE_300, system.serial.BIT_RATE_600, system.serial.BIT_RATE_1200, system.serial.BIT_RATE_2400, system.serial.BIT_RATE_4800, system.serial.BIT_RATE_9600, system.serial.BIT_RATE_19200, system.serial.BIT_RATE_38400, system.serial.BIT_RATE_57600, system.serial.BIT_RATE_115200, system.serial.BIT_RATE_230400, system.serial.BIT_RATE_460800, system.serial.BIT_RATE_921600
```

Integer dataBits - Configure the data bits. Valid values are defined by the following constants [optional]:

```
system.serial.DATA_BITS_5, system.serial.DATA_BITS_6, system.serial.DATA_BITS_7, system.serial.DATA_BITS_8
```

Integer handshake - Configure the handshake. Valid values are defined by the following constants [optional]:

```
system.serial.HANDSHAKE_CTS_DTR, system.serial.HANDSHAKE_CTS_RTS, system.serial.HANDSHAKE_DSR_DTR, system.serial.HANDSHAKE_HARD_IN, system.serial.HANDSHAKE_HARD_OUT, system.serial.HANDSHAKE_NONE, system.serial.HANDSHAKE_SOFT_IN, system.serial.HANDSHAKE_SOFT_OUT, system.serial.HANDSHAKE_SPLIT_MASK, system.serial.HANDSHAKE_XON_XOFF
```

Boolean hardwareFlowControl - Configure hardware flow control. On or off. [optional]

Integer parity - Configure parity. Valid values are defined by the following constants [optional]:

```
system.serial.PARITY_EVEN, system.serial.PARITY_ODD, system.serial.PARITY_MARK, system.serial.PARITY_SPACE, system.serial.PARITY_NONE
```

Integer stopBits - Configure stop bits. Valid values are defined by the following constants [optional]:

```
system.serial.STOP_BITS_1, system.serial.STOP_BITS_2
```

- Returns

SerialConfigurator - A SerialConfigurator object with exposed functions that can be used to configure the serial port instead of, or in addition to, the arguments passed to `configureSerialPort`.

- Scope

Gateway, Vision Client, Perspective Session

SerialConfigurator Methods

Below is a listing of methods on the SerialConfigurator object. All methods return the original SerialConfigurator object, but with a modified parameter value. For a list of possible values, see the appropriate parameter on the function description above.

Method	
setBitRate	Sets the bit rate on the SerialConfigurator.
setDataBits	Sets the data bits on the SerialConfigurator.
setParity	Sets the parity on the SerialConfigurator.
setStopBits	Sets the stop bits on the SerialConfigurator.
setFlowControl	Sets the flow control on the SerialConfigurator.
setHandshake	Sets the handshake on the SerialConfigurator.
setHardwareFlowControl	Sets the hardware flow control on the SerialConfigurator.

Code Examples

Code Snippet - Configuring Serial Port

```
# Configure a serial port using keyword args.  
# The "port" keyword is mandatory.  
  
system.serial.configureSerialPort(\  
    port="COM1", \  
    bitRate=system.serial.BIT_RATE_9600, \  
    dataBits=system.serial.DATA_BITS_8, \  
    handshake=system.serial.HANDSHAKE_NONE, \  
    hardwareFlowControl=False, \  
    parity=system.serial.PARITY_NONE, \  
    stopBits=system.serial.STOP_BITS_1)
```

Code Snippet - Configuring Serial Port

```
# Configure a serial port using a SerialConfigurator (returned by configureSerialPort()):  
  
system.serial.configureSerialPort("COM1") \  
    .setBitRate(system.serial.BIT_RATE_9600) \  
    .setDataBits(system.serial.DATA_BITS_8) \  
    .setHandshake(system.serial.HANDSHAKE_NONE) \  
    .setHardwareFlowControl(False) \  
    .setParity(system.serial.PARITY_NONE) \  
    .setStopBits(system.serial.STOP_BITS_1)
```

Keywords

```
system serial configureSerialPort, serial.configureSerialPort
```

system.serial.openSerialPort

This function is used in [Python Scripting](#).

Description

Opens a previously configured serial port for use. Will throw an exception if the serial port cannot be opened.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.openSerialPort(port)

- Parameters

String port - The name of the serial port, e.g., "COM1" or "dev/ttyS0".

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system serial openSerialPort, serial.openSerialPort

system.serial.port

This function is used in **Python Scripting**.

Description

Returns a [context manager](#) wrapping a serial port, allowing the rest of the system to interact with that port. This function effectively combines the [system.serial.configureSerialPort](#), [system.serial.openSerialPort](#), and [system.serial.closeSerialPort](#) functions into a single call.

Intended to be used with the [Python 'with' statement](#). The object aliased in the 'with' statement has special access to all of the other [system.serial](#) functions, allowing for reads and writes.

Closing the port happens automatically once the 'with' statement ends.

Accepts the same arguments as [configureSerialPort](#), and access to constants must be prefixed by "system.serial." (as shown in the parameter descriptions).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.port(port, [bitRate], [dataBits], [handshake], [hardwareFlowControl], [parity], [stopBits])

- Parameters

[String](#) port - The name of the serial port, e.g., "COM1" or "dev/ttyS0".

[Integer](#) bitRate - Configure the bit rate. Valid values are defined by the following constants [optional]:

```
system.serial.BIT_RATE_110, system.serial.BIT_RATE_150, system.serial.BIT_RATE_300, system.  
serial.BIT_RATE_600, system.serial.BIT_RATE_1200, system.serial.BIT_RATE_2400, system.serial.  
BIT_RATE_4800, system.serial.BIT_RATE_9600, system.serial.BIT_RATE_19200, system.serial.  
BIT_RATE_38400, system.serial.BIT_RATE_57600, system.serial.BIT_RATE_115200, system.serial.  
BIT_RATE_230400, system.serial.BIT_RATE_460800, system.serial.BIT_RATE_921600
```

[Integer](#) dataBits - Configure the data bits. Valid values are defined by the following constants (optional):

```
system.serial.DATA_BITS_5, system.serial.DATA_BITS_6, system.serial.DATA_BITS_7, system.  
serial.DATA_BITS_8
```

[Integer](#) handshake - Configure the handshake. Valid values are defined by the following constants [optional]:

```
system.serial.HANDSHAKE_CTS_DTR, system.serial.HANDSHAKE_CTS_RTS, system.serial.  
HANDSHAKE_DSR_DTR, system.serial.HANDSHAKE_HARD_IN, system.serial.HANDSHAKE_HARD_OUT, system.  
serial.HANDSHAKE_NONE, system.serial.HANDSHAKE_SOFT_IN, system.serial.HANDSHAKE_SOFT_OUT,  
system.serial.HANDSHAKE_SPLIT_MASK, system.serial.HANDSHAKE_XON_XOFF
```

[Boolean](#) hardwareFlowControl - Configure hardware flow control on or off. [optional]

[Integer](#) parity - Configure parity. Valid values are defined by the following constants [optional]:

```
system.serial.PARITY_EVEN, system.serial.PARITY_ODD, system.serial.PARITY_MARK, system.serial.  
PARITY_SPACE, system.serial.PARITY_NONE
```

[Integer](#) stopBits - Configure stop bits. Valid values are defined by the following constants [optional]:

```
system.serial.STOP_BITS_1, system.serial.STOP_BITS_2
```

- Returns

[PortManager](#) - A wrapper around the configured port, that can be entered by using a 'with' statement. The port will automatically close on exiting the 'with' statement scope.

- Scope

Gateway, Vision Client, Perspective Session

Using the PortManager

The PortManager is the primary way to interact with a serial port when using this function. It has special access to the other system serial functions. Specifically:

- `system.serial.readBytes`
- `system.serial.readBytesAsString`
- `system.serial.readLine`
- `system.serial.readUntil`
- `system.serial.sendBreak`
- `system.serial.write`
- `system.serial.writeBytes`

Calling these functions from the PortManager does **not** require the 'port' parameter, as the port is implied by `system.serial.port`. However all other parameters are available (see the linked pages in the bullet list above).

In addition, you do not include 'system.serial.' when accessing the other serial functions mentioned above, as the aliased object has access to them. Thus:

```
# Correct
with system.serial.port("COM1") as port:
    port.write("some string")

# Incorrect
with system.serial.port("COM1") as port:
    system.serial.write("COM1", "some string")
```

Code Examples

Example 1: Simple Example with Descriptions

```
# Reads a value from a port.

# First we call the function using a 'with' statement, and create an aliased object named 'port'
with system.serial.port("COM1", bitRate=system.serial.BIT_RATE_9600) as port:

    # Within the 'with' statement, we can call other serial functions by referencing the aliased
    object.
        # Meaning, in this example, 'port' can easily call the system.serial.readLine() function with the
    following:
        line = port.readLine(60000)
```

Example 2: Using all Parameters

```
# Same idea as example one, but uses all available parameters.
with system.serial.port(
    port = "COM1",
    bitRate = system.serial.BIT_RATE_110,
    dataBits = system.serial.DATA_BITS_5,
    handshake = system.serial.HANDSHAKE_CTS_DTR,
    hardwareFlowControl = False,
    parity = system.serial.PARITY_EVEN,
    stopBits = system.serial.STOP_BITS_1) as port:

    line = port.readLine(60000)
```

Keywords

`system serialport, serial.port`

system.serial.readBytes

This function is used in [Python Scripting](#).

Description

Read `numberOfBytes` bytes from a serial port.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.serial.readBytes(port, numberOfBytes, [timeout])`

- Parameters

[String](#) `port` - The previously configured serial port to use.

[Integer](#) `numberOfBytes` - The number of bytes to read.

[Integer](#) `timeout` - Maximum amount of time, in milliseconds, to block before returning. Default is 5000. [optional]

- Returns

[List\[Byte\]](#) - A list containing bytes read from the serial port.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

`system serial readBytes, serial.readBytes`

system.serial.readBytesAsString

This function is used in [Python Scripting](#).

Description

Read `numberOfBytes` bytes from a serial port and convert them to a String. If a specific encoding is needed to match the source of the data, use [sys stem.serial.readBytes](#) and use the desired encoding to decode the byte array returned.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.readBytesAsString(port, numberOfBytes, [timeout], [encoding])

- Parameters

[String](#) `port` - The previously configured serial port to use.

[Integer](#) `numberOfBytes` - The number of bytes to read.

[Integer](#) `timeout` - Maximum amount of time, in milliseconds, to block before returning. Default is 5000. [optional]

[String](#) `encoding` - Encoding to use when constructing the string. Defaults to the platform's default character set. [optional]

- Returns

[String](#) - A String created from the bytes read.

- Scope

Gateway, Vision Client, Perspective Session

The Encoding Parameter

The encoding parameter can be used to decode a string with any of the possible encoding character sets that are available. By default, the following character sets are provided by [the Java platform](#) (dash characters and underscores are interchangeable). Dashed examples are shown below:

- ISO-8859-1
- US-ASCII
- UTF-16
- UTF-16BE
- UTF-16LE
- UTF-8

Code Examples

There are no code examples for this function.

Keywords

system serial readBytesAsString, serial.readBytesAsString

system.serial.readLine

This function is used in **Python Scripting**.

Description

Attempts to read a line from a serial port. A "line" is considered to be terminated by either a line feed ('\n'), a carriage return ('\r'), or a carriage return followed immediately by a line feed.

The function will wait until the timeout period for a terminator. If the timeout is reached before the line is properly terminated, then the buffer will be dumped, possibly resulting in data loss.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.readLine(port, [timeout], [encoding])

- Parameters

String port - The previously configured serial port to use.

Integer timeout - Maximum amount of time, in milliseconds, to block before returning. Default is 5000. [optional]

String encoding - The String encoding to use. Default is UTF8. [optional]

- Returns

String - A line of text.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no examples associated with this scripting function.

Keywords

system serial readLine, serial.readLine

system.serial.readUntil

This function is used in **Python Scripting**.

Description

Reads a byte at a time from a serial port until a delimiter character is encountered. The read will block for up to timeout milliseconds before returning.

Caution: If the delimiter is not found before the timeout period, then the buffer will dump, potentially resulting in data loss.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.readUntil(port, delimiter, includeDelimiter, [timeout])

- Parameters

[String](#) port - The previously configured serial port to use.

[Char](#) delimiter - The delimiter to read until.

[Boolean](#) includeDelimiter - If true, the delimiter will be included in the return value.

[Integer](#) timeout - Timeout in milliseconds. Default is 5000. [optional]

- Returns

[String](#) - Returns a string containing all 8-bit ASCII characters read until the delimiter was reached, and including the delimiter if the "includeDelimiter" parameter was true.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system serial readUntil, serial.readUntil

system.serial.sendBreak

This function is used in [Python Scripting](#).

Description

Sends a break signal for approximately millis milliseconds.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.sendBreak(port, millis)

- Parameters

String port - The name of the serial port, e.g., "COM1" or "dev/ttyS0".

Integer millis - Approximate length of break signal, in milliseconds.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system serial sendBreak, serial.sendBreak

system.serial.write

This function is used in [Python Scripting](#).

Description

Write a string to a serial port using the platforms default character encoding.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.write(port, toWrite, [encoding])

- Parameters

[String](#) port - The previously configured serial port to use.

[String](#) toWrite - The string to write.

[String](#) encoding -Encoding to decode the string with, for example: UTF-8. Default is the platform default character set. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

The encoding Parameter

The encoding parameter can be used to decode a string with any of the possible encoding character sets that are available. By default, the following character sets are provided by [the Java platform](#) (dash characters and underscores are interchangeable. Dashed examples are shown below:

- ISO-8859-1
- US-ASCII
- UTF-16
- UTF-16BE
- UTF-16LE
- UTF-8

Code Examples

There are no examples associated with this scripting function.

Keywords

system serial write, serial.write

system.serial.writeBytes

This function is used in [Python Scripting](#).

Description

Write a List[Byte] to a serial port.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.serial.writeBytes(port, toWrite)

- Parameters

[String](#) port - The previously configured serial port to use.

[List\[Byte\]](#) toWrite - The List[Byte] to write.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system serial writeBytes, serial.writeBytes

system.sfc

SFC Functions

The following functions give you access to interact with the SFCs in the Gateway.

In This Section ...

Chart Scope Variables

There are a number of built-in variables maintained by the SFC engine that can be read through the `chart` scope.

Note: Certain chart scoped variables may interfere with the internal functions of the chart. For example, creating a variable like `chart.values` will conflict with a `pyDictionary`'s `values()` method and therefore the chart will show an error. Since SFC charts use Python Dictionaries to manage chart scoped variables the methods associated with Python Dictionary's act like reserved words.

Variable Name	Description
<code>chart.instanceId</code>	The string UUID of the running chart instance
<code>chart.startTime</code>	A <code>java.util.Date</code> object that indicates when the chart instance started running.
<code>chart.runningTime</code>	An integer representing the number of seconds the chart has been running for.
<code>chart.parent</code>	The chart scope of the enclosing chart (if any). <code>null</code> if this chart was not executed as part of an enclosing step.
<code>chart.running</code>	Returns true if the chart is in the running state
<code>chart.state</code>	An integer representing the state of the chart as the following: 0 Aborted 1 Aborting 2 Cancelled 3 Canceling 4 Initial 5 Paused 6 Pausing 7 Resuming 8 Running 9 Starting 10 Stopped 11 Stopping

system.sfc.cancelChart

This function is used in **Python Scripting**.

Description

Cancels the execution of a running chart instance. Any running steps will be told to stop, and the SFC chart will enter Canceling state. Will throw a KeyError if the ID does not match any running chart instance.

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.cancelChart(id)

- Parameters

String id -The ID of the chart instance to cancel.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following attempts to stop an SFC but will alert the user if the id of the chart is not currently
running.
id = 'Some long string value. It can be obtained using system.sfc.getRunningCharts()'
try:
    system.sfc.cancelChart(id)
except:
    system.gui.messageBox("Could not stop the SFC")
```

Keywords

system sfc cancelChart, sfc.cancelChart

system.sfc.getRunningCharts

This function is used in **Python Scripting**.

Description

Retrieves information about running charts. Can search all running charts, or be filtered charts at a specific path. This function will return charts that are in a paused state.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.sfc.getRunningCharts([chartPath])

- Parameters

String chartPath - The path to a chart to filter on: i.e., "folder/chartName". If specified, only charts at the path will be included in the returned dataset. If omitted, the function will return data for all active charts.

- Returns

Dataset - A dataset with information on the active chart. Contains the following columns:

- instanceId - The chart instance, or UUID of the chart.
- chartPath - The path to the chart.
- startDate - A date object noting when the chart instance started.
- startedBy - The name of the user that started the chart.
- chartState - The current state of the chart. Possible states are "Running" and "Paused".
- keyParamName - Name of the chart's key parameter. Returns None if a key parameter is not defined.
- keyParamValue - Value of the chart's key parameter. Returns None if a key parameter is not defined.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Example - Check All Running Charts

```
# This example checks for all running charts, and return a formatted string detailing each chart instance.

# Check for all running charts. The path may be specified as a string to filter the results.
data = system.sfc.getRunningCharts()
# Create a string to append chart data to. The "\n" is a new line character.
chartData = "The following charts are running:\n"

# Iterate through each chart.
for row in range(data.rowCount):

    # Extract the instanceId and chartPath values from the current row.
    runningChartId = data.getValueAt(row, "instanceId")
    runningChartPath = data.getValueAt(row, "chartPath")

    # Append a string to chartData with the values extracted above.
    chartData += "Id: %s, Path: %s\n" % (runningChartId, runningChartPath)

# Print the string of chart Id's and paths.
print chartData
```

Example - Retrieve Chart instanceId Using chartPath

```
# This example will return the instanceId of chart instances with a specific chartPath.  
# A valid path must be defined for this example.  
  
# Return data for running instances at a specific path. "folder/myChart" should be replaced with a valid  
path.  
data = system.sfc.getRunningCharts("folder/myChart")  
  
# Initialize a list to contain all instance IDs  
chartIds = []  
  
# Iterate through each chart, and fetch the instanceId  
for row in range(data.rowCount):  
    chartIds.append(data.getValueAt(row, "instanceId"))  
  
# Print the chartIds list  
print chartIds
```

Keywords

system sfc getRunningCharts, sfc.getRunningCharts

system.sfc.getVariables

This function is used in **Python Scripting**.

Description

Get the variables in a chart instance's scope. Commonly used to check the value of a chart parameter, or determine how long the chart has been running for.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.sfc.getVariables(instanceld)

- Parameters

[String](#) instanceld - The instance identifier of the chart.

- Returns

[PyChartScope](#) - Effectively a Python dictionary of variables, Step scopes for active steps are found under the "activeSteps" key. In addition to those keys, Chart Parameters will also be included in the dictionary as keys. More information on this object can be found in our [Javadocs](#).

- Scope

Gateway, Vision Client, Perspective Session

Keys in the PyChartScope

PyChartScope Description

The following keys are in the PyChartScope object.

Key	Description	Value Type								
parent	If the chart is enclosed in another chart, this Dictionary returns information on the parent chart. Otherwise, returns None. The keys returned by the parent dictionary is identical to calling system.sec.getVariables() directly on an instance of the parent chart.	Dictionary								
instanceld	The instance identifier of the chart.	Unicode								
startTime	A date object representing when the chart started.	Date								
runningTime	A long representing the amount of time the chart has been running.	Long								
chartPath	A path (as shown in the Project Browser) leading to the chart.	String								
activeSteps	A dictionary of all active steps in the chart. The keys in this dictionary are UUID values representing the individual steps. The value of each key, is another dictionary, with the following keys	Dictionary								
	<table border="1"><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>The step's ID.</td></tr><tr><td>name</td><td>The name of the active step.</td></tr><tr><td>runningTime</td><td>The amount of time (as a long) that the step has been active.</td></tr></tbody></table>	Key	Value	id	The step's ID.	name	The name of the active step.	runningTime	The amount of time (as a long) that the step has been active.	
Key	Value									
id	The step's ID.									
name	The name of the active step.									
runningTime	The amount of time (as a long) that the step has been active.									

Chart Params	In addition to the built-in keys mentioned above, each configured chart parameter will be represented as a key: value pair in the PyChartScope.	Varies, based on the value of the chart parameter
--------------	---	---

Code Examples

Example - Show Chart Data to the User

```
"""
This example will show the chart path and start time of a single chart in a messageBox.
We can make use of the SFC Monitor component to give the users the ability to pick a single
running chart
"""

# Fetch the ID of a running chart. In this case, we used the Instance ID property on a SFC Monitor
# component
id = event.source.parent.getComponent('SFC Monitor').instanceId

# Retrieve the variables from the chart
chartVars = system.sfc.getVariables(id)

# Show the path and starttime of the chart in a messageBox
system.gui.messageBox("Chart Path: %s has been running since %s" % (chartVars["chartPath"], chartVars
["startTime"]))

```

Example - Print the name and running time for all active steps

```
# Get the name and running time for each step in each running chart.

# Return data for running instances at a specific path. "folder/myChart" should be replaced with a valid
path.
data = system.sfc.getRunningCharts("folder/myChart")

# Initialize a list to contain all instance IDs
chartIds = []
# Iterate through each chart, and fetch the instanceId
for row in range(data.rowCount):
    chartIds.append(data.getValueAt(row, "instanceId"))

# Now that we have the ID for all active charts, pull variables out of each.
for id in chartIds:
    chartVars = system.sfc.getVariables(id)

    # Prints the chart instance ID. In the context of this example, this line is used to delineate
    # between all our print statements.
    print "Details for Chart ID: %s" % chartVars["instanceId"]

    # Create a variable that references the activeSteps dictionary. Creating a variable here
    # makes the syntax below a bit cleaner.
    allSteps = chartVars["activeSteps"]

    # Iterate through the active steps. A "step" represents the key of each step
    # in the activeSteps ("allSteps") dictionary
    for step in allSteps:

        # store the value of the current step dictionary in a variable. This is simply to keep
        # the syntax below clean. Equivalent to: chartVars["activeSteps"]
[step]
        currStep = allSteps[step]
```

```
# Print out the name and running time of each step.  
print "Step %s has been running for %i seconds" % (currStep['name'], currStep  
['runningTime'])
```

Keywords

system sfc getVariables, sfc.getVariables

system.sfc.pauseChart

This function is used in **Python Scripting**.

Description

Pauses a running chart instance. Any running steps will be told to pause, and the chart will enter paused state. Will throw a `KeyError` if the ID does not match any running chart instance.

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.pauseChart(id)

- Parameters
 - String `id` - The ID of the chart instance to pause
- Returns
 - Nothing
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system sfc pauseChart, sfc.pauseChart

system.sfc.redundantCheckpoint

This function is used in **Python Scripting**.

Description

Synchronizes chart and step variables of the specified chart instance across a redundant cluster, allowing the chart instance to continue where it left off if a redundant failover occurs. Check out [redundancy sync](#) for more information.

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.redundantCheckpoint(instanceId)

- Parameters

 String instanceId - The instance identifier of the chart.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system sfc redundantCheckpoint, sfc.redundantCheckpoint

system.sfc.resumeChart

This function is used in **Python Scripting**.

Description

Resumes a chart that was paused. Steps which were previously paused will be resumed, and chart will enter resuming state. Will throw a `KeyError` if the ID does not match any running chart instance.

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.resumeChart(id)

- Parameters
 - String `id` - The ID of the chart instance to resume.
- Returns
 - Nothing
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function

Keywords

system sfc resumeChart, sfc.resumeChart

system.sfc.setVariable

This function is used in **Python Scripting**.

Description

Sets a variable inside a currently running [SFC chart](#).

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.setVariable(instanceId, [stepId], variableName, variableValue)

- Parameters

String instanceId - The instance identifier of the chart.

String stepId - The id for a step inside of a chart. If omitted the function will target a chart scoped variable.] [optional]

String variableName - The name of the variable to set.

Object variableValue - The value for the variable to be set to.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session



Omitting the **stepId** parameter will cause the function to target a chart scoped variable. If the variable is persistent to the whole chart, or used in multiple different steps, then this parameter should be omitted.

If a stepId parameter is used, then the function will target a step scoped variable. The step associated with the stepId must be the currently active step.

Code Examples

Code Snippet

```
# The following Action step script passes the chart instance ID and step ID to a client message Handler.  
The message handler can then wait  
# for user input, and then write back to the step variables.  
  
# The example assumes there is a chart scoped variable called confirmEndChart, and a step scoped variable  
# called "messageSent".  
  
# Get the instanceId of the current chart.  
chartID = chart.get("instanceId")  
  
# Get the id of the step.  
stepID = step.get("id")  
  
# Create a payload to pass to the client.  
# Include the instanceId and stepId so the script from the message handler knows which  
# chart and step to write to.
```

```

payload = {"chartID" : chartID, "stepID" : stepID}

# Send the message.
system.util.sendMessage(project = "SFC", messageHandler = "SFCMessages", payload = payload)

#####
# The following script would be placed on a client message handler. This receives the payload,
# and sets a variable on either the chart or step depending on user selection

# Read items out of the payload.
id = payload['chartID']
stepId = payload['stepID']

# Ask the user to end the chart.
if system.gui.confirm("Would you like to end the process"):
    #If yes, end the chart. confirmEndChart is chart scoped, so only 3 parameters are passed
    system.sfc.setVariable(id,"confirmEndChart",True)
else:
    #If no, reset the step.messageSent variable so that the user will be prompted again.
    #messageSent is step scoped, so 4 parameters are passed
    system.sfc.setVariable(id,stepId,"messageSent",False)

```

Keywords

system sfc setVariable, sfc.setVariable

system.sfc.setVariables

This function is used in **Python Scripting**.

Description

Sets any number of variables inside a currently running chart.

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.setVariables(instanceId, [stepId], variableMap)

- Parameters

String instanceId - The instance identifier of the chart.

String stepId - The id for a step inside of a cart. If omitted the function will target a chart scoped variable. [optional]

Dictionary[String, Any] variableMap - A dictionary containing the name:value pairs of the variables to set.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session



Omitting the **stepId** parameter will cause the function to target a chart scoped variable. If the variable is persistent to the whole chart, or used in multiple different steps, then this parameter should be omitted.

If a stepId parameter is used, then the function will target a step scoped variable. The step associated with the stepId must be the currently active step.

Code Examples

Code Snippet

```
# Get the instance ID from the selected chart on a SFC Monitor component.  
id = event.source.parent.getComponent('SFC Monitor').instanceId  
  
# Create a Python dictionary of values. This example assumes there are variables on the  
# chart named chartParam and counter. The script will set these to 1, and 0 respectively.  
dict = {"chartParam":1, "counter":0}  
  
# Set the variables on the chart.  
system.sfc.setVariables( id, dict)
```

Keywords

system sfc setVariables, sfc.setVariables

system.sfc.startChart

This function is used in **Python Scripting**.

Description

Starts a new instance of an SFC chart. The chart must be set to "Callable" execution mode.

Client Permission Restrictions

Permission Type: SFC Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.sfc.startChart(path, arguments)

- Parameters

String projectName - The name of the project that the chart was created in.

String chartPath - The path to the chart, for example "ChartFolder/ChartName".

Dictionary[String, Any] parameters - A dictionary of arguments. Each key-value pair in the dictionary becomes a variable in the chart scope and will override any default.

- Returns

String - The unique ID of this chart.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following starts an SFC with a dictionary of values to use inside the chart.  
args= {"var1":10, "Var2":15, "Var3":1}  
path = "ChartFolder/ChartName"  
sfcID = system.sfc.startChart("MyProject", path, args)
```

Keywords

system sfc startChart, sfc.startChart

system.tag

Tag Functions

The following functions give you access to interact with Ignition Tags.

[In This Section ...](#)

system.tag.browse

This function is used in [Python Scripting](#).

Description

Returns a list of [nodes](#) found at the specified path. The list objects are returned as dictionaries with some basic information about each node.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag. browse(path, filter)

- Parameters

[String](#) path - The path that will be browsed, typically to a folder or UDT instance.

[Dictionary\[String, Any\]](#) filter - A dictionary of browse filter keys. Keys are listed below.

- Returns

[Results](#) - A Results object which contains a list of Tag dictionaries, one for each Tag found during the browse. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Filter Keys

The following keys represent filter criteria that can be used by the `filter` parameter.

Key	Description	Example Filter
name	The name of the item. Utilizes the * character as a wildcard character.	<pre># Literally search for "MyTag" {"name": "MyTag"} # Searches for any names that contain "Tag" in their name {"name": "*Tag*"}</pre>
data Type	Represents the data type on the Tag. Valid values can be found on the Tag Properties page .	<pre>{"dataType": "Int4"}</pre>
value Source	Represents how the node derives its value. Generally only used by nodes with a Tag type of "AtomicTag". Valid values can be found on the Value Source description on the Tag Properties page .	<pre>{"valueSource": "opc"}</pre>
tag Type	The type of the node (Tag, folder, UDT instance, etc). A list of possible types can be found on the Tag Properties page .	<pre>{"tagType": "AtomicTag"}</pre>

typeId	Represents the UDT type of the node. If the node is a UDT definition, then the value will be <code>None</code> . If the node is not a UDT, then this filter choice will not remove the element. As such, this filter functions best when paired with a tagType filter with a value of <code>UdtInstance</code> .	<pre>{"typeId": "myUDT", "tagType": "UdtInstance"}</pre>
quality	Represents the quality on the node. While there are many types of quality codes, this function only recognizes "Bad" and "Good". More granular quality codes are ignored.	<pre>{"quality": "Good"}</pre>
maxResults	Limits the amount of results that will be returned by the function.	<pre>{"maxResults": 10}</pre>
recursive	<p>This feature is new in Ignition version 8.1.2 Click here to check out the other new features</p> <p>Allows the browse to find all Tags inside the starting folder or UDT instance, even if they are inside nested folders or UDT instances themselves. Accepted values are True and False. False is the default, meaning that the browse will only return tags directly inside the starting folder or UDT instance.</p>	<pre>{"recursive": True}</pre>

Results Object

The contents of each dictionary in the Results object varies based on the tagType of the node in question.

General Keys

By default all dictionaries contain the following:

Key	Description
fullPath	<p>A fully qualified Tag path to the node, including the name of the node.</p> <p>The value returned by this key is a <code>BasicTagPath</code>. However you can easily cast the variable to a string:</p> <pre># Browse the Tag Provider named "default". results = system.tag/browse("[default]") for i in results: stringPath = str(i['fullPath']) # Do something useful with the stringPath...</pre>
hasChildren	A boolean representing if the node contains sub-nodes, such as folders and UDT definitions. Useful in cases where you need to recursively call the <code>browse</code> function.
name	The name of the node.
tagType	The type of the node.

Tag Keys

If the node is a Tag (`tagType = AtomicTag`), then it will also contain the following keys:

Key	Description
<code>dataType</code>	The data type of the Tag.
<code>valueSource</code>	Represents how the Tag derives its value.
<code>value</code>	The last known qualified value on the Tag.

UDT Keys

Both UDT Instances and UDT Definitions add the following key:

Key	Description
<code>tagType</code>	Represents the type the UDT is based off of. UDT Definitions will have a value of <code>None</code> .

Code Examples

Code Snippet - Simple Browse

```
# This simple script will browse a given Tag path, in this case the root of the provider called default, and print the results.

results = system.tag/browse(path = '[default]', filter = {})
for result in results.getResults():
    print result
```

Code Snippet - Filtered Browse

```
# This simple script will browse a given Tag path, in this case the root of the provider called default, and print the results.
# It also is filtering out anything that is not Atomic Tag, like folders and UDT Instances.

results = system.tag/browse(path = '[default]', filter = {'tagType':'AtomicTag'})
for result in results.getResults():
    print result
```

Code Snippet - Wildcards with the Name Parameter

```
# Similar to the Filtered Browse above, except a wildcard character may be used when filtering on the name parameter
# The wildcard character (the * character) represents any number of characters, including none.

results = system.tag/browse(path = '[default]', filter = {'name':'*M*'})
for result in results.getResults():
    print result
```

Code Snippet - Simple Browse with Condition

```
# This simple script will browse a given Tag path, in this case the root of the Tag Provider called default, and print the results.
# After it browses, it finds all of the items that do not have children and prints only those.
```

```
results = system.tag.browse(path = '[default]', filter = {})
for result in results.getResults():
    if result['hasChildren'] == False:
        print result
```

Code Snippet - Recursive Browse

```
# This script has created a browseTags function which can be called with a Tag path and filter.
# The function will recursively find all items under that path by going into folders and UDT Instances.
# This example gives the initial path of '[default]', meaning it will find every item in the Tag Provider
# called default.
```

```
results = system.tag.browse("[default]", {"tagType":"UdtInstance", "recursive":True}).results
for result in results:
    print str(result['fullPath'])
```

Keywords

system tag browse, tag.browse

system.tag/browseHistoricalTags

This function is used in [Python Scripting](#).

Description

Will browse for any historical Tags at the provided historical path. It will only browse for Tags at the path, and will not go down through any children. Will return with a BrowseResults object, which can be accessed using the methods below:

- .getResults() will get the underlying resultset.
- .getReturnedSize() will get the number of records in the resultset.
- .getContinuationPoint() will get the continuation point if this function was limited, allowing you to use it in another function call to continue the browse.

The resultset returned from .getResults() is a list of Results objects. This list can be iterated through with a standard for loop, and each object in the list can be accessed with the following methods:

- .getPath() will get the full Historical Tag Path for that object.
- .getType() will get the type of the object.
- .hasChildren() a flag indicating whether or not the object has any children.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag/browseHistoricalTags(path, nameFilters, maxSize, continuationPoint)

- Parameters

String path - The Historical Tag Path to browse. See the Tag Export page for a [description of how to construct a historical Tag Path](#).

List[String] nameFilters - A list of name filters to be applied to the result set.

Integer maxSize - The maximum size of the result set.

Any continuationPoint - Sets the continuation point in order to continue a browse that was previously started and then limited. Use getContinuationPoint() on the Results object (see *Returns* below) to get the continuation point.

- Returns

Results - A Results object which contains a list of Tag dictionaries, one for each Tag found during the browse. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This script will browse for any history tags at the specified historical path and print out all of their Historical Tag Paths to the console.

path='histprov:DB:/drv:controller:default:/tag:simulator/turbine 3'
browse = system.tag/browseHistoricalTags(path) #We call the function and place the BrowseResults that get returned into a variable called browse.
results = browse.getResults() #We can then call getResults() on the BrowseResults variable, and store that in a variable called results.
for result in results: #We can now loop through the results in a for loop.
    print result.getPath() #We then call .getPath() on the individual objects to get the Tag Path.
```

Caution: The following script can be very dangerous, as it recursively calls itself until there are no more children. If you have a lot of Historical Tags and provide it with a path that is to something on the top level, it could take a long time and even lock up your system.

```
# This script will browse for Historical Tags and print their Historical Tag Path to the console,
# starting from the specified path,
# and going all the way down until there are no more children.
# This is useful because the function by itself will only provide results that are located at the
# specified path, but not for anything further in.
# This function recursively calls itself if there are any results that still have children.
# So if the specified path has any folders, the function will browse those as well until it can't browse
# any further.
# If you have a lot of Historical Tags and do not specify a path in the function, it will browse for all
# of your Historical Tags,
# which could take some time and may lock up your system. It is recommended to specify some sort of path.

def browse(path='histprov:DB:/drv:controller:default:/tag:simulator'):
    for result in system.tag.browseHistoricalTags(path).getResults():
        print result.getPath()
        if result.hasChildren():
            browse(result.getPath())
browse()
```

Keywords

system tag browseHistoricalTags, tag.browseHistoricalTags

system.tag.configure

This function is used in [Python Scripting](#).

Description

Creates Tags from a given list of Python dictionaries or from a JSON source string. Can be used to overwrite a current Tag's configuration.

When utilizing this function, the Tag definitions must specify the names of properties with their scripting/JSON name. A reference of these properties can be found on the [Tag Properties](#) and [Tag Alarm Properties](#) pages.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.tag.configure(basePath, tags, [collisionPolicy])

- Parameters

String basePath - The starting point where the new Tags will be created. When making changes to existing Tags with this function, you want to set the path to the parent folder of the exiting Tag(s), not the Tag(s) themselves.

Any tags - A list of Tag definitions, where each Tag definition is a Python dictionary. Alternately, a JSON source string may be passed to this parameter. When editing existing Tags, it is generally easier to retrieve the Tag configurations with [system.tag.getConfiguration](#), modify the results of the getConfiguration call, and then write the new configuration to the parent folder of the existing Tag(s).

String collisionPolicy - The action to take when a Tag or folder with the same path and name is encountered. Defaults to Overwrite. [optional]. Possible values include:

- a - Abort and throw an exception
- o - Overwrite and replace existing Tag's configuration
- i - Ignore that item in the list.
- m - merge, modifying values that are specified in the definition, without impacting values that aren't defined in the definition. Use this when you want to apply a slight change to Tags, without having to build a complete configuration object.

- Returns

List - A List of QualityCode objects, one for each Tag in the list, that is representative of the result of the operation. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Python - Edit Multiple Tags

```
# This example will retrieve some existing Tag configurations, make changes to the configurations,
# and write the new configurations to the original Tags.

# Define a base path. Sometime useful, but we could hardcode the paths
# in the getConfiguration() and configure() calls.
parentPath = "[NewProvider]SomeFolder"

# Get the current configurations recursively, which is useful when targeting
# folders and UDT Instances.
configs = system.tag.getConfiguration(parentPath + "/AnotherFolder", True)

# The getConfiguration() above always returns a list of dictionaries, and our results
```

```

# are inside of the first dictionary.
for tag in configs[0]['tags']:

    # Check each node. At this point we may get folders or other nodes
    # that we're uninterested in. Examine tagType to figure it out.
    # Note that we're making sure the tagType is in fact a string for this comparison.
    if str(tag['tagType']) != 'Folder':

        # Make a change to the Tag. If we're iterating over UDT instances,
        # then we can change the parameters here. Commented example below:
        # tag['parameters']['myParam'] = foo

        # In our case, we'll just disable the Tags.
        tag['enabled'] = False

system.tag.configure(parentPath, configs, "o")

```

Python - Adding a New Tag

```

# This example will add a new OPC Tag. It can be further expanded to modify more
# properties on the Tag. Additionally, this example can be used to edit an existing Tag
# by setting the baseTagPath to a Tag that already exists, and by modifying the collision policy.

# The provider and folder the Tag will be placed at.
baseTagPath = "[default]MyFolder"

# Properties that will be configured on that Tag.
tagName = "myNewTag"
opcItemPath = "ns=1;s=[Simulator]_Meta:Sine/Sine0"
opcServer = "Ignition OPC-UA Server"
valueSource = "opc"
sampleMode = "TagGroup"
tagGroup = "Default"

# Configure the tag.
tag = {
    "name": tagName,
    "opcItemPath" : opcItemPath,
    "opcServer": opcServer,
    "valueSource": valueSource,
    "sampleMode" : sampleMode,
    "tagGroup" : tagGroup
}

# Set the collision policy to Abort. Thus, if a Tag already exists at the base path,
# we will not override the Tag. If you are overwriting an existing Tag, then set this to "o".
collisionPolicy = "a"

# Create the Tag.
system.tag.configure(baseTagPath, [tag], collisionPolicy)

```

Python - Interacting with Alarms

```

# The provider and folder the Tag will be placed at.
baseTagPath = "[default]"

# Create a list of alarms, where each alarm is a Python dictionary.
alarms = [
    {
        "name": "My scripting alarm",
        "mode": "AboveValue",
        "setpointA": 10
    }
]

# Configure the list of Tags. We're only interacting with a single Tag, but still need to pass
# a list as an argument.

```

```

tags = [
    {
        "alarms":alarms,
        "name":"myTag"
    }
]

# Abort if this example attempts to overwrite any of your existing Tags.
collisionPolicy = "a"

# Create the Tag.
system.tag.configure(baseTagPath, tags, collisionPolicy)

```

Python - Add UDT Instance

```

# This example will add a new UDT Instance. It can be further expanded to modify more
# properties on the Tag. Additionally, this example can be used to edit an existing Tag
# by setting the baseTagPath to a Tag that already exists, and by modifying the collision policy.

# The provider and folder the Tag will be placed at.
baseTagPath = "[default]Motors"

# Properties that will be configured on that Tag.
tagName = "Motor 1"
typeId = "Motor"
tagType = "UdtInstance"
# Parameters to pass in.
motorNum = "1"

# Configure the Tag.
tag = {
    "name": tagName,
    "typeId" : typeId,
    "tagType" : tagType,
    "parameters" : {
        "motorNum" : motorNum
    }
}

# Set the collision policy to Abort. That way if a tag already exists at the base path,
# we will not override the Tag. If you are overwriting an existing Tag, then set this to "o".
collisionPolicy = "a"

# Create the Tag.
system.tag.configure(baseTagPath, [tag], collisionPolicy)

```

Python - Adding Folders in other Folders

```

# Folders are nodes with a 'tagType' set to 'Folder'.
# Each folder can contain a 'tags' value, which contains other tags and folders.

Tags={'tagType': 'Folder',
      'name': 'NewFolderName',
      'tags' : [
          {
              'name': 'anotherfolder',
              'tagType': 'Folder',
              'tags': []
          }
      ]
}

system.tag.configure(
    basePath = '',
    tags = Tags,
    collisionPolicy = "o"
)

```

Python - Writing to Parameters in UDT Definition

In this example, we're going to change the value on a UDT Definition parameter with a script. It assumes there is a UDT Definition already configured at the root of the Data Types folder (in this case, named "myUdtDef"), and contains a parameter (named "myParam").

The screenshot shows two windows. The top window is the 'Tag Browser' with a toolbar and a tree view of tags. The 'Tags' node has a child 'Data Types' node, which contains a 'myUdtDef' entry. The bottom window is the 'Tag Editor' titled 'myUdtDef > Parameters'. It shows a table with one row for 'myParam'. Below the table are 'Commit' and 'Revert' buttons.

Name	Value
myParam	

```
# UDT Definitions reside in the "_types_" folder, which can be retrieved
# via the Tag Browser : right-click > Copy Tag Path.
# Retrieving the existing configuration is much easier than typing it all out.
tag = system.tag.getConfiguration("[default]_types_/_myUdtDef")

# This line is accessing the first tag in our results (the UDT Definition), then returns the
# 'parameters' dictionary, which then provides access to individual parameters.
tag[0]['parameters']['myParam'] = '300'

# Overwrite the existing configuration.
collisionPolicy = "o"

# Write the new configuration to our existing UDT Definition.
# Note that the first parameter is to the parent folder of the Definition,
# not a path to the Definition.
system.tag.configure("[default]_types_", tag, collisionPolicy)

# Once the configure call finishes, myParam on the Definition should have a value of 300.
```



Keywords

system tag configure, tag.configure

system.tag.copy

This function is used in **Python Scripting**.

Description

Copies Tags from one folder to another. Multiple Tag and folder paths may be passed to a single call of this function. The new destination can be a separate Tag provider.

Copying UDTs Across Tag Providers

When copying UDTs to a different provider, the destination provider must have a matching UDT definition. The function copies Tags sequentially, so definitions can be placed earlier in the list, with instances following after:

```
# The '_types_' part of the path denotes the Data Types folder. You can retrieve the path to your UDT
# definition with right-click on
# on the Tag in the Tag Browser > Copy Tag Path.
udtDef = '[default]_types_/Motor'
udtInstances = '[default]UDT_Instances_Folder'

# When building the Tag list, place the definitions in list before the instances.
tags = [udtDef, udtInstances]
```

Tag Groups

Tag Groups will not be copied to the new provider, so the copied Tags may not initially execute. This can be remedied by creating a matching Tag Group in the destination provider, either before or after the Tags have been copied.

Remote Tag Providers

This function can move Tags to or from a [Remote Tag provider](#). In this case, the Tag Access [Service Security](#) settings on both providers must be set to ReadWriteEdit.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.tag.copy(tags, destination, [collisionPolicy])

- Parameters

[List](#) tags - A List of Tag paths to move.

[String](#) destination - The destination to copy the Tags to. All specified Tags will be copied to the same destination. The destination Tag provider must be specified.

[String](#) collisionPolicy - The action to take when a Tag or folder with the same path and name is encountered. Possible values include: "a" Abort and throw an exception, "o" Overwrite and replace existing Tag's configuration, "i" Ignore that item in the list. Defaults to Overwrite. [optional]

- Returns

[List](#) - A List of QualityCode objects, one for each Tag in the list, that is representative of the result of the operation. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Copy a Tag or Folder

```
# Define the tag/folder path(s).
tags = ['[default]Old_Tags/Subfolder' ]

# We'll move the tag/folder above to the new path.
destination = '[default]New_Tags'

# If there is a collision with the destination, we'll abort the process.
policy = 'a'

# Copy the Tags.
system.tag.copy(tags, destination, policy)
```

Keywords

system tag copy, tag.copy

system.tag.deleteAnnotations

This feature is new in Ignition version **8.1.0**
[Click here](#) to check out the other new features

This function is used in **Python Scripting**.

Description

Removes stored annotations from the sqllite_annotations table. Requires the full Tag path (including history provider) for each annotation, as well as each annotation's storage ID.

The function expects two lists (PySequences) of equal length. The items in each list is 1-to-1, meaning the first item in the "paths" list relates to the first item in the "storageIds" list, the second item in "paths" relates to the second item in "storageIds", etc.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.deleteAnnotations(paths, storageIds)

- Parameters

List[String] paths - A list of Tag paths with existing annotations. The paths are equivalent to what would be used for a Tag history query, and should specify the source provider as well. For example, "[HistoryProvider/Gateway:Provider]Path/To/Tag".

List[String] storageIds - A sequence of storage identifiers that will be deleted. Storage ID values can be retrieved with [system.tag.queryAnnotations](#).

- Returns

A list of qualified values. The quality code will indicate success or failure, and if successful, the storage id of the annotation will have been deleted. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Clients, Perspective Sessions

Code Examples

Code Snippet

```
paths = ["[My-Provider]Station_1/ph", "[My-Provider]Station_2/ph"]
storageId = [2,3]

system.tag.deleteAnnotations(paths, storageId)
```

Keywords

system tag deleteAnnotations, tag.deleteAnnotations

system.tag.deleteTags

This function is used in **Python Scripting**.

Description

Deletes multiple Tags or Tag Folders. When deleting a Tag Folder, all Tags under the folder are also deleted.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when ran from the Gateway scope.

Syntax

system.tag.deleteTags(tagPaths)

- Parameters

[List](#) tagPaths - A List of the paths to the Tags or Tag Folders that are to be removed.

- Returns

[List](#) - A List of QualityCode objects, one for each Tag in the list, that is representative of the result of the operation. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Deleting Tags

```
# This example will delete several Tags. Modify the paths argument to change which Tags will be deleted.  
# Note that a confirmation isn't automatically included, so be cautious when calling this on a production  
server.  
  
paths = [  
    '[default]A_Tag',  
    '[default]Folder/Some_Other_Tag'  
]  
  
# Delete the Tags.  
results = system.tag.deleteTags(paths)  
  
# We could expand this example further by examining the list of quality codes...  
for index in range(len(results)):  
  
    # ...check if a returned anything except Good.  
    if results[index].isNotGood():  
  
        # ...and do something if we failed, such as retrieve the Tag path from earlier, and pair  
        # it with a quality code.  
        print 'Could not delete tag at path: %s \n Reason: %s' % (paths[index], results[index])
```

Keywords

system tag deleteTags, tag.deleteTags

system.tag.exists

This function is used in [Python Scripting](#).

Description

Checks whether or not a Tag with a given path exists.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.exists(tagPath)

- Parameters

[String](#) tagPath - The path of the Tag to look up.

- Returns

[Boolean](#) - True if a Tag exists for the given path, false otherwise.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would write a 1 to the Tag "Compressors/C28/ClearFault" if that Tag exists.
```

```
if system.tag.exists("Compressors/C28/ClearFault"):  
    system.tag.write("Compressors/C28/ClearFault", 1)
```

Keywords

system tag exists, tag.exists

system.tag.exportTags

This function is used in [Python Scripting](#).

Description

Exports Tags to a file on a local file system.

The term "local file system" refers to the scope in which the script was running; for example, running this script in a Gateway Timer script will export the file to the Gateway file system.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.exportTags(filePath, tagPaths, [recursive], exportType)

- Parameters

[String](#) filePath - The file path that the Tags will be exported to. If the file does not already exist, this function will attempt to create it.

[List](#) tagPaths - A List of Tag paths to export. All Tag paths in the list must be from the same parent folder.

[Boolean](#) recursive - Set to true to export all Tags under each Tag path, including Tags in child folders. Defaults to true. [optional]

[String](#) exportType - The type of file that will be exported. Set to "json" or "xml". Defaults to "json".

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example attempts to export the entire Tag Provider, including UDT definitions.

# The filepath is Operating System dependent, so the path notation below may differ based on where the
# function is called from.
filePath = 'C:\\\\Users\\\\myUser\\\\Desktop\\\\myTags'
tagPaths = ["[default]"]
recursive = True

system.tag.exportTags(filePath, tagPaths, recursive)
```

Keywords

system tag exportTags, tag.exportTags

system.tag.getConfiguration

This function is used in [Python Scripting](#).

Description

Retrieves Tags from the Gateway as Python dictionaries. These can be edited and then saved back using [system.tag.configure](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.getConfiguration(basePath, recursive)

- Parameters

String basePath - The starting point where the Tags will be retrieved. This can be a folder containing, and if recursive is true, then the function will attempt to retrieve all of the Tags in the folder.

Boolean recursive - If true, the entire Tag tree under the specified path will be retrieved.

- Returns

List - A List of Tag dictionaries. Nested Tags are placed in a list marked as "tags" in the dictionary.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Python - Access a Single Property on a Single Tag

```
# This example will look up the Data Type property on a Tag.  
  
path = '[default]My_Folder/My_Tag'  
  
config = system.tag.getConfiguration(path, False)  
  
# While the call above was directed at a single Tag, the function  
# still returns a list, so we access index 0 to examine the properties  
# (hence the "[0]").  
#  
# Additionally, we can access the DataType property in a similar manner  
# to accessing a key in a Python Dictionary.  
  
print config[0]['dataType']
```

Python - Get All Properties for a Single Configuration

```
# This example will get the configuration of a single Tag  
  
# Update the path here with the Tag path you're trying to reach  
path = '[default]Sine/Sine0'  
  
# Get the configurations  
tags = system.tag.getConfiguration(path)
```

```

for tagDict in tags:

    # Iterate over the dictionary with the iteritems function
    for key, value in tagDict.iteritems():

        # Do something with the keys and values
        print key, ' : ', value

```

Python - Return an Entire Folder of Tag Configurations

```

# This example will get the configurations of Tags under a folder.

# Update the path here with the folder you want to start at
folder = '[default]Folder/Another_Folder'

# Get the configurations. We'll specify True for the second parameter to search
# recursively
nodes = system.tag.getConfiguration(folder, True)

# Iterate over the results
for item in nodes:

    # Through the results, search each dictionary
    for key, value in item.iteritems():

        # ...looking for a 'tags' key
        if key == 'tags':
            print #####Found some tags!#####

        # iterate over the Tag configurations we found
        for tagConfig in value:

            # Do something with the results.
            print tagConfig["name"]

```

Keywords

system tag getConfiguration, tag.getConfiguration

system.tag.importTags

This function is used in **Python Scripting**.

Description

Imports a JSON Tag file at the provided path. Also supports XML and CSV Tag file exports from legacy systems.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.tag.importTags(filePath, basePath, [collisionPolicy])

- Parameters

String filePath - The file path of the Tag export to import.

String basePath - The Tag path that will serve as the root node for the imported Tags.

String collisionPolicy - The action to take when a Tag or folder with the same path and name is encountered. Possible values include: "a" Abort and throw an exception, "o" Overwrite and replace existing Tag's configuration, "i" Ignore that item in the list. Defaults to Overwrite. [optional]

- Returns

List - A List of QualityCode objects, one for each Tag in the list, that is representative of the result of the operation. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system tag importTags, tag.importTags

system.tag.isOverlaysEnabled

This function is used in [Python Scripting](#).

Description

Returns whether or not the current client's quality overlay system is currently enabled.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.isOverlaysEnabled()

- Parameters
 - Nothing
- Returns
 - [Boolean](#) - True if overlays are currently enabled.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system tag isOverlaysEnabled, tag.isOverlaysEnabled

system.tag.move

This function is used in [Python Scripting](#).

Description

Moves Tags or Folders to a new destination. The new destination can be a separate Tag provider. If interested in copying the Tags to a new destination, instead of moving them, please see the [system.tag.copy](#) page.

Moving Across Tag Providers

When moving UDTs to a different provider, the destination provider must have a matching UDT definition. This function moves folders/Tags sequentially, so definitions can be placed earlier in the list, with instances following after.

Note that moving Tags with this function will not move or otherwise update prior Tag History or Alarm Journal entries: the old records will persist in the database at the old path, while future entries will be based on the new paths.

Remote Tag Providers

Additionally, this function can move Tags to or from a [Remote Tag provider](#). In this case, the Tag Access Service Security settings on both providers must be set to ReadWriteEdit.

Client Permission Restrictions

[Permission Type](#): Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.tag.move(tags, destination, [collisionPolicy])

- Parameters

[List](#) tags - A List of Tag paths to move.

[String](#) destination - The destination to move the Tags to. The destination Tag provider must be specified: i.e., [default]Folder /myTag.

[String](#) collisionPolicy - The action to take when a Tag or folder with the same path and name is encountered. Possible values include: "a" Abort and throw an exception, "o" Overwrite and replace existing Tag's configuration, "i" Ignore that item in the list. Defaults to Overwrite. [optional]

- Returns

[List](#) - A List of QualityCode objects, one for each Tag in the list, that is representative of the result of the operation. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Rename a Folder

```
# This function moves a folder in a Tag provider to a new folder.  
  
# Since both paths are at the same node, the "Old_Folder" will be placed at the root of the provider, and  
# the name changed, effectively renaming the folder.  
tags = ['[default]Old_Folder']  
destination = '[default]New_Folder/'
```

```
# Move the folder.  
system.tag.move(tags, destination)
```

Code Snippet - Move Multiple Folders to a New Folder

```
# This function moves an entire folder of Tags to a new destination.  
  
# Define the source and new path.  
tags = ['[default]Old_Folder', '[default]Another_Folder' ]  
destination = '[default]New_Folder/Imports'  
  
# If there are any conflicts with the new destination, then abort the operation  
policy = 'a'  
  
# Move the folder  
system.tag.move(tags, destination)
```

Keywords

system tag move, tag.move

system.tag.queryAnnotations

This feature is new in Ignition version **8.1.0**.
[Click here](#) to check out the other new features

This function is used in **Python Scripting**.

Description

Queries annotations stored in the Tag History system for a set of paths for a given time range.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.queryAnnotations(paths, [startTime], [endTime], [types])

- Parameters

List[String] paths - A list of Tag paths to query. The paths are equivalent to what would be used for a Tag History Query, and should specify the Source Providers as well. For example, "[HistoryProvider/Gateway:Provider]Path/To/Tag".

Date startTime - The start of the time range. If not defined, defaults to 12 hours ago. [optional]

Date endTime - The end of the time range. If not defined, defaults to "now". [optional]

List[String] types - A list of string "types" to filter on. Types are defined by the annotations and various subsystems, and may vary with different providers. Possible annotation types are listed on the [system.tag.storeAnnotations](#) page. [optional]

- Returns

List[Annotation] A list of [Annotation](#) objects that match the query criteria.

- Scope

Gateway, Vision Clients, Perspective Sessions

The Annotation Object

Properties on the annotation objects returned by this function can be references by name (i.e., Annotation.storageId). The table below represent properties on the object.

Property	Description
path	Represents the tag path associated with the annotation.
type	Represents the type of annotation.
rangeStart	The start time of the annotation.
rangeEnd	The end time of the annotation.
data	Any data (such as a description, or user entered text) associated with the annotation.
storageId	Represents the ID value of the annotation, as listed in the sqllt_annotations table. Used in conjunction with the system.tag.storeAnnotations to change existing annotations, and system.tag.deleteAnnotations to remove annotations.
deleted	Flag representing if the entries are deleted or not.

Code Examples

Code Snippet

```
paths = ["[My-Provider]Station_1/ph"]
annotations = system.tag.queryAnnotations(paths)

for i in annotations:
    print i.storageID
```

Keywords

system tag queryAnnotations, tag.queryAnnotations

system.tag.queryTagCalculations

This function is used in [Python Scripting](#).

Description

Queries various calculations (aggregations) for a set of Tags over a specified range. Returns a dataset with a row per Tag, and a column per calculation.

This is useful when you wish to aggregate Tag history collected over a period of time into a single value per aggregate. If you want multiple values aggregated to a single time slice (i.e., hourly aggregates for the same Tag over an 8 hour period) consider using [system.tag.queryTagHistory](#)

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax



This function accepts [keyword arguments](#).

system.tag.queryTagCalculations(paths, calculations, [startDate], [endDate], [rangeHours], [rangeMinutes], [aliases], [includeBoundingValues], [validatesSCExec], [noInterpolation], [ignoreBadQuality])

- Parameters

List[String] paths - An array of Tag paths (strings) to query calculations for. The resulting dataset will have a row for each Tag, and a column for each calculation.

List[String] calculations - An array of calculations (aggregation functions) to execute for each Tag. Valid values are: "Average" (time-weighted), "MinMax", "LastValue", "SimpleAverage", "Sum", "Minimum", "Maximum", "DurationOn", "DurationOff", "CountOn", "CountOff", "Count", "Range", "Variance", "StdDev", "PctGood", and "PctBad".

Date startDate - The starting point for the calculation window. If omitted, and range is not used, 8 hours before the current time is used. [optional]

Date endDate - The end of the calculation window. If omitted, and range is not used, uses the current time. [optional]

Integer rangeHours - Allows you to specify the query range in hours, instead of using start and end date. Can be positive or negative, and can be used in conjunction with startDate or endDate. [optional]

Integer rangeMinutes - Same as rangeHours, but in minutes. [optional]

List[String] aliases - Aliases that will be used to override the Tag path names in the result dataset. Must be 1-to-1 with the Tag paths. If not specified, the Tag paths themselves will be used. [optional]

Boolean includeBoundingValues - A boolean flag indicating that the system should attempt to load values before and after the query bounds for the purpose of interpolation. The effect depends on the aggregates used. The default is true. [optional]

Boolean validatesSCExec - A boolean flag indicating whether or not data should be validated against the scan class execution records. If false, calculations may include data that is assumed to be good, even though the system may not have been running. Default is true. [optional]

Boolean noInterpolation - A boolean flag indicating that the system should not attempt to interpolate values in situations where it normally would, such as for analog Tags. Default is false. [optional]

Boolean ignoreBadQuality - A boolean flag indicating that bad quality values should not be used in the query process. If set, any value with a "bad" quality will be completely ignored in calculations. Default is false. [optional]

- Returns

Dataset - A dataset representing the calculations over the specified range. A demonstration of the table appears below. There is a row per Tag id, and a column per requested calculation. Tag path is returned in the first column.

tagpath	calculation1	calculation2	calculationN
path1	value	value	value

path2	value	value	value
pathN	value	value	value

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
system.tag.queryTagCalculations(paths=['Historical Tag'], calculations=['Average'],
noInterpolation=False)
```

Code Snippet

```
# Build a list of String tag paths
paths = [
            "[default]Folder/Tag1",
            "[default]Folder/Tag2"
        ]

# Determine the calculation to use
calc = ["StdDev"]

# Define the date range
end = system.date.now()
start = system.date.parse("2019-07-30 4:00:00")

# Run the query, returning the results as an Ignition dataset
data = system.tag.queryTagCalculations(paths, calc, start, end)

# From here you would need to do something useful with the data variable. You could extract the values
# and write them to a Tag, pass them to a dataset property on a component, or any number of other
things.
print "The calculated value for the first tag is " + str(data.getValueAt(0,1))
print "The calculated value for the second tag is " + str(data.getValueAt(1,1))
```

Keywords

system tag queryTagCalculations, tag.queryTagCalculations

system.tag.queryTagDensity

This function is used in [Python Scripting](#).

Description

Queries the Tag history system for information about the density of data. In other words, how much data is available for a given time span.

This function is called with a list of Tag paths, and a start and end date. The result set is a two column dataset specifying the timestamp, and a relative weight. Each row is valid from the given time until the next row. Tags are assigned a 1 or a 0 if they are present or not. All values are then multiplied together to get a decimal based percentage for the density. Thus, for four Tag paths passed in, if three Tags were present during the span, the result would be 0.75.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.queryTagDensity(paths, startDate, endDate)

- Parameters

[List\[String\]](#) paths - An array of Tag paths (strings) to query.

[Date](#) startDate - The start of the range to query.

[Date](#) endDate - The end of the range to query.

- Returns

[Dataset](#) - A two-column dataset consisting of a timestamp and a weight. Each row is valid until the next row.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Print Example

```
# Will grab the density of a Tag and print it out to the console.
density = system.tag.queryTagDensity(
    ['[default]myTag'],
    system.date.addHours(system.date.now(), -24),
    system.date.now())

density = system.dataset.toPyDataSet(density)
for row in density:
    print row[0], row[1]
```

Density Readouts for Each Day

```
# Create a list of times going back in time at day increments.
# This forces density readouts at each day, even if two days had the same density.
now = system.date.now()
times = [system.date.addDays(now, -1), system.date.addDays(now, -2), system.date.addDays(now, -3), system.
date.addDays(now, -4), system.date.addDays(now, -5), system.date.addDays(now, -6)]

# Create start and end date variables, as well as a variable that holds each history density.
startDate = now
endDate = now
```

```

list = []

# Loop through the list of times.
for time in times:

    # Set the new end date to whatever the start date was previously
    # and the new start date to the next time in the list.
    endDate = startDate
    startDate = time

    # Query Tag Density using a list of Tagpaths with the startDate and endDate values.
    density = system.tag.queryTagDensity(
        ['[default]tag1', '[default]tag2', '[default]tag3', '[default]tag4', '[default]tag5'],
        startDate, endDate)

    # Add each row of the returned dataset to a list of rows.
    density = system.dataset.toPyDataSet(density)
    for row in density:
        list.append([row[0], row[1]])

# Place the results in a table.
event.source.parent.getComponent('Table').data = system.dataset.toDataSet(['Times', 'Density Percentages'], list)

```

Code Snippet

```

# Create a list of times going back in time at day increments.
# This forces density readouts at each day, even if two days had the same density.
# This differs from the previous example in that it uses the new system.dataset.appendDataset function,
# which is only available in 7.9.7.
now = system.date.now()
times = [system.date.addDays(now, -1), system.date.addDays(now, -2), system.date.addDays(now, -3), system.
date.addDays(now, -4), system.date.addDays(now, -5), system.date.addDays(now, -6)]

# Create start and end date variables.
startDate = now
endDate = now

# Loop through the list of times.
for time in times:
    # Set the new end date to whatever the start date was previously
    # and the new start date to the next time in the list.
    endDate = startDate
    startDate = time

    # Query Tag Density using a list of Tagpaths with the startDate and endDate values.
    density = system.tag.queryTagDensity(
        ['[default]EquipmentFour', '[default]Ramp/Ramp6', '[default]Ramp/Ramp7', '[default]Ramp/Ramp8',
        '[default]Ramp/Ramp9'],
        startDate, endDate)
    if endDate == now:
        densities = density
    else:
        densities = system.dataset.appendDataset(densities, density)

# Place the results in a table.
event.source.parent.getComponent('Table').data = densities

```

Keywords

system tag queryTagDensity, tag.queryTagDensity

system.tag.queryTagHistory

This function is used in [Python Scripting](#).

Description

Issues a query to the Tag Historian. Querying Tag history involves specifying the Tags and the date range, as well as a few optional parameters. The Tag historian will find the relevant history and then interpolate and aggregate it together into a coherent, tabular result set.

This is useful when you're trying to retrieve Tag history data over a period of time (i.e., multiple timeslices over a period of time). If you are trying to take a range of time and aggregate the data to a single value then consider using [system.tag.queryTagCalculations](#).

This function takes a list of strings, where each string is a Tag path, like "Tanks/Tank5" or "[OracleProvider]Sump/Out2". See also: [Tag Paths](#).

The return size determines how the underlying data is aggregated and/or interpolated. If a distinct return size is specified, that will be the number of rows in the resulting dataset. The special numbers 0 and -1 mean "Natural" and "On-Change", respectively. "Natural" calculates a return size based on the rate of the logging historical scan classes. For example, if you query 1 hour of data for a scan class logging every minute, the natural return size is 60. "On-Change means that you'll get an entry whenever any of the Tags under consideration have changed.

Instead of defining a fixed return size, the parameters intervalHours and intervalMinutes can be used. These parameters can be used independently or together to define a "window size". For example, if you defined a 1 hour range, with intervalMinutes=15, you would get 4 rows as a result.

The span of the query can be specified using startDate and endDate. You can also use rangeHours and rangeMinutes in conjunction with either start or end date to specify the range in dynamic terms. For example, you could specify only "rangeHours=-8" to get the last 8 hours from the current time. Or you could use "startDate='2012-05-30 00:00:00', rangeHours=12" to get the first half of the day for May 30th, 2012. The aggregation mode is used when the data is denser than what you asked for. This happens when using fixed return sizes, as there will often be multiple raw values for the window interval defined. Another common operation is to set the return size to 1, in order to use these aggregate functions for calculation purposes. The available functions are:

- "MinMax" - Will return two entries per time slice - the min and the max.
- "Average" - Will return the time-weighted average value of all samples in that time slice.
- "LastValue" - Returns the most recent actual value to the end of the window. Note that if a value does not exist in this window, a 0 will be returned in cases where interpolation is turned off.
- "SimpleAverage" - Returns the simple mathematical average of the values - $((V_1+V_2+\dots+V_n)/n)$
- "Maximum" - The maximum value of the window.
- "Minimum" - The minimum value of the window.
- "DurationOn" - The time, in seconds, that a value has been boolean true.
- "DurationOff" - The time, in seconds, that a value has been boolean false.
- "CountOn" - The number of times the value has transitioned to boolean true.
- "CountOff" - The number of times the value has transitioned to boolean false.
- "Count" - The number of "good", non-interpolated values per window.
- "Range" - The difference between the min and max.
- "Variance" - The variance for "good", non-interpolated values. Does not time weight.
- "StdDev" - The standard deviation for "good", non-interpolated values. Does not time weight.
- "PctGood" - The percentage of time the value was good.
- "PctBad" - The percentage of time the value was bad.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax



This function accepts [keyword arguments](#).

```
system.tag.queryTagHistory(paths, [startDate], [endDate], [returnSize], [aggregationMode], [returnFormat], [columnNames],  
[intervalHours], [intervalMinutes], [rangeHours], [rangeMinutes], [aggregationModes], [includeBoundingValues], [validateSCExec],  
[noInterpolation], [ignoreBadQuality], [intervalSeconds], [rangeSeconds])
```

- Parameters

`List[String]` paths - An array of Tag paths (strings) to query. Each Tag path specified will be a column in the result dataset.

`Date` startDate - The earliest value to retrieve. If omitted, 8 hours before current time is used. [optional]

`Date` endDate - The latest value to retrieve. If omitted, current time is used. [optional]

`Integer` returnSize - The number of samples to return. -1 will return values as they changed, and 0 will return the "natural" number of values based on the logging rates of the scan class(es) involved. -1 is the default. [optional]

`String` aggregationMode - The mode to use when aggregating multiple samples into one time slice. Valid values are: "Average" (time-weighted), "MinMax", "LastValue", "SimpleAverage", "Sum", "Minimum", "Maximum", "DurationOn", "DurationOff", "CountOn", "CountOff", "Count", "Range", "Variance", "StdDev", "PctGood", and "PctBad". [optional]

`String` returnFormat - Use "Wide" to have a column per Tag queried, or "Tall" to have a fixed-column format. Default is "Wide". [optional]

`List[String]` columnNames - Aliases that will be used to override the column names in the result dataset. Must be 1-to-1 with the Tag paths. If not specified, the Tag paths themselves will be used as column titles. [optional]

`Integer` intervalHours - Allows you to specify the window interval in terms of hours, as opposed to using a specific return size. [optional]

`Integer` intervalMinutes - Same as intervalHours, but in minutes. Can be used on its own, or in conjunction with intervalHours. [optional]

`Integer` rangeHours - Allows you to specify the query range in hours, instead of using start and end date. Can be positive or negative, and can be used in conjunction with startDate or endDate. [optional]

`Integer` rangeMinutes - Same as rangeHours, but in minutes. [optional]

`List[String]` aggregationModes - A one-to-one list with paths specifying an aggregation mode per column. [optional]

`Boolean` includeBoundingValues - A boolean flag indicating that the system should attempt to include values for the query bound times if possible. The default for this property depends on the query mode, so unless a specific behavior is desired, it is best to not include this parameter. [optional]

`Boolean` validateSCExec - A boolean flag indicating whether or not data should be validated against the scan class execution records. If false, data will appear flat (but good quality) for periods of time in which the system wasn't running. If true, the same data would be bad quality during downtime periods. [optional]

`Boolean` noInterpolation - A boolean flag indicating that the system should not attempt to interpolate values in situations where it normally would. This will also prevent the return of rows that are purely interpolated. [optional]

`Boolean` ignoreBadQuality - A boolean flag indicating that bad quality values should not be used in the query process. If set, any value with a "bad" quality will be completely ignored in calculations and in the result set. [optional]

`Integer` timeout - Timeout in milliseconds for Client Scope. This property is ignored in the Gateway Scope. [optional]

This feature is new in Ignition version **8.1.0**

[Click here](#) to check out the other new features

`Integer` intervalSeconds - Same as intervalHours and interval Minutes, but in seconds. Can be used on its own, or in conjunction with intervalHours and intervalMinutes. [optional]

`Integer` rangeSeconds - Allows you to specify the query range in seconds, instead of using start and end date. Can be positive or negative, and can be used in conjunction with startDate or endDate. [optional]

- Returns

`Dataset` - A dataset representing the historian values for the specified Tag paths. The first column will be the timestamp, and each column after that represents a Tag.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following example will return a dataset with one row detailing maximum value of a Tag named 'Sine' for the past 30 minutes.
endTime = system.date.now()
startTime = system.date.addMinutes(endTime, -30)
dataSet = system.tag.queryTagHistory(paths=['Sine'], startDate=startTime, endDate=endTime, returnSize=1, aggregationMode="Maximum", returnFormat='Wide')
```



Keywords
system tag queryTagHistory, tag.queryTagHistory

system.tag.readAsync

This function is used in [Python Scripting](#).

Description

Asynchronously reads the value of the Tags at the given paths. You must provide a python callback function that can process the read results.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.readAsync(tagPaths, callback)

- Parameters

[List](#) tagPaths - A List of Tag paths to read from. If no property is specified in the path, the Value property is assumed.

[Callable](#) callback - A Python callback function to process the read results. The function definition must provide a single argument, which will hold a List of qualified values when the callback function is invoked. The qualified values will have three sub-members: value, quality, and timestamp.

- Returns

Nothing

- Scope

Gateway, Vision Clients, Perspective Sessions

Code Examples

Code Snippet

```
# Define a function that will iterate over the results of our async read.
def checkValues(asyncReturn):

    # In this case we'll just create a counter, and increment it when values are over 100.
    counter = 0
    for qValue in asyncReturn:
        if qValue.value > 100:
            counter += 1

    # This part may not work depending on scope (Vision Client/Designer).
    # Replace this part of the function with something more useful, such as
    # a Tag or DB write.
    system.gui.messageBox(str(counter))

# Define the Tag paths you want to read.
paths = [
            "[default]Tag1",
            "[default]Tag2",
        ]

system.tag.readAsync(paths, checkValues)
```

Keywords

system tag readAsync, tag.readAsync

system.tag.readBlocking

This function is used in [Python Scripting](#).

Description

Reads the value of the Tags at the given paths. Will block the script from continuing until the read operation is complete or times out.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.readBlocking(tagPaths, [timeout])

- Parameters

[List](#) tagPaths - A List of Tag paths to read from. If no property is specified in a path, the Value property is assumed.

[Integer](#) timeout - How long to wait in milliseconds before the read operation times out. This parameter is optional, and defaults to 45000 milliseconds if not specified. [optional]

- Returns

[List](#) - A list of QualifiedValue objects corresponding to the Tag paths. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Read From Multiple Tags

```
# Specify the paths.
paths = [
    "[default]Folder/Tag_A",
    "[default]Folder/Tag_B"
]

# Send the reads off.
values = system.tag.readBlocking(paths)

# Here we can examine each value.
for i in range(len(values)):
    print "Tag at Path: %s\n Had a value of %s" % (paths[i], values[i].value)
```

Keywords

system tag readBlocking, tag.readBlocking

system.tag.requestGroupExecution

This function is used in [Python Scripting](#).

Description

Sends a request to the specified Tag Group to execute now.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.tag.requestGroupExecution(tagPath)
```

- Parameters
 - [String provider](#) - Name of the Tag Provider that the Tag group is in.
 - [String tagGroup](#) - The name of the Tag group to execute.
- Returns
 - Nothing
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system tag requestGroupExecution, tag.requestGroupExecution

system.tag.setOverlaysEnabled

This function is used in **Python Scripting**.

Description

Enables or disables the component quality overlay system.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.setOverlaysEnabled(enabled)

- Parameters
 - Boolean enabled - True to turn on Tag overlays; false to turn them off.
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

There are no code examples for this function.

Keywords

system tag setOverlaysEnabled, tag.setOverlaysEnabled

system.tag.storeAnnotations

This feature is new in Ignition version **8.1.0**
[Click here](#) to check out the other new features

This function is used in **Python Scripting**.

Description

Stores annotations into the tag history system. Annotations are stored by the underlying historian implementations, so different providers may store in different ways, and some providers may not support annotation storage. All parameters are 1-to-1, so all provided lists should be of the same length. If a particular annotation doesn't need a parameter, that element can be None in the list.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.tag.storeAnnotations(paths, startTimes, [endTimes], [types], [data], [storageIds], [deleted])

- Parameters

List[String] paths - A list of tag paths to store for. The paths are equivalent to what would be used for a tag history query, and should specify the source provider as well. For example, "[HistoryProvider/Gateway:Provider]Path/To/Tag". This parameter is required, even if storage ids are included, because it is used to identify the underlying storage provider.

List[Date] startTimes - The start times of the events. If omitted, defaults to the current time.

List[Date] endTimes - The end times of the event, if applicable. If omitted, does not store an end time for the annotation. [optional]

List[Annotation] types - The type id for the annotation. If not defined, "marker" will be used. See the [Annotation Types](#) for more details. [optional]

List[String] data - Data for the annotation, such as text describing the meaning of the annotation. [optional]

List[String] storageIds - If defined, the function will instead update the existing annotation instead of adding new ones, overriding existing values for the annotation with those provided by this function (if the corresponding delete parameter is True). Storage id is available on the Annotation object, and is returned as the result value from the storeAnnotations call. [optional]

List[Boolean] deleted - A list of booleans indicating that the individual annotation should be deleted. Requires storage id to be set as well. [optional]

- Returns

A list of QualifiedValues. The quality code will indicate success or failure. If successful, the storage id of the annotation will be returned in the value. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Clients, Perspective Sessions

Annotation Types

Possible annotation types are listed below.

Type	Description
note	A user created annotation. Used in cases specifically where a user adds the annotation (such as from the Power Chart's built-in interface).
range	A date range that indicates a special area.
marker	A singular point in time.
trace	A single point or date range that indicates an X-trace.

Code Examples

Example - Minimal Arguments

```
paths = [[My-Provider]Station_1/ph]
system.tag.storeAnnotations(paths)
```

Example - More Arguments

```
paths = [[My-Provider]Station_1/ph]
startTimes = [system.date.now()]
endTimes = [system.date.addMinutes(startTimes[0], 1)]
types = ["range"]
data = ["some text"]

system.tag.storeAnnotations(paths, startTimes, endTimes, types, data)
```

Example - Updating Existing Annotations

```
# First, retrieve existing annotations based on some criteria.
paths = [[My-Provider]Station_1/ph]
time = system.date.now()
annotations = system.tag.queryAnnotations(paths, system.date.addMinutes(time,-10), time)

# Iterate over the annotations, checking for some other criteria. In this case, we'll change "marker"
# types to "note" types.
for a in annotations:
    if a.type == "marker":
        system.tag.storeAnnotations([a.path], [a.rangeStart], [a.rangeEnd], ["note"], [a.data],
[a.storageId])
```

Keywords

system tag storeAnnotations, tag.storeAnnotations

system.tag.writeAsync

This function is used in **Python Scripting**.

Description

Asynchronously writes values to Tags at the given paths. You must provide a Python callback function that can process the write results.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.tag.writeAsync(tagPaths, values, [callback])

- Parameters

[List](#) tagPaths - A List of Tag paths to write to. If no property is specified in a Tag path, the Value property is assumed.

[List](#) values - The values to write to the specified paths.

[Callable](#) callback - A Python callback function to process the write results. The function definition must provide a single argument, which will hold a List of quality codes when the callback function is invoked. The quality codes will hold the result of the write operation for that Tag. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no scripting examples for this function.

Keywords

system tag writeAsync, tag.writeAsync

system.tag.writeBlocking

This function is used in **Python Scripting**.

Description

Writes values to Tags at the given paths. This function will block the script from continuing until the write operation is complete or times out.

Client Permission Restrictions

Permission Type: Tag Editing

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.tag.writeBlocking(tagPaths, values, [timeout])

- Parameters

[List](#) tagPaths - A List of Tag paths to write to. If no property is specified in a Tag path, the Value property is assumed.

[List](#) values - A list of values to write to the specified Tag paths.

[Integer](#) timeout - How long to wait in milliseconds before the write operation times out. This parameter is optional and defaults to 45000 milliseconds if not specified. [optional]

- Returns

[List](#) - A List of QualityCode objects, one for each Tag path. See [Scripting Object Reference](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Write to Multiple Tags

```
# Specify the paths.
paths = [
    "[default]Folder/Tag_A",
    "[default]Folder/Tag_B"
]

# Specify the values.
values = [
    1,
    2
]
# Send the writes off.
system.tag.writeBlocking(paths, values)
```

Keywords

system tag writeBlocking, tag.writeBlocking

system.twilio

Tag Functions

The following functions give you access to read info and send SMS through Twilio. This requires the [Twilio module](#), which is not included in a typical install.

[In This Section ...](#)

system.twilio.getAccounts

This function is used in [Python Scripting](#).

Description

Return a list of Twilio accounts that have been configured in the Gateway.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.twilio.getAccounts()

- Parameters
 - Nothing
- Returns
 - [List](#) - A list of configured Twilio accounts.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Retrieves a list of Twilio accounts and then iterates through the resulting list.  
# Call system.twilio.getAccounts() and store the returned list into a variable.  
twilioAccounts = system.twilio.getAccounts()  
  
# Iterate through the list of accounts.  
for account in twilioAccounts:  
  
    # Prints the account name to the console, but could do something more useful with each account.  
    print account
```

Keywords

system twilio getAccounts, twilio.getAccounts

system.twilio.getAccountsDataset

This function is used in [Python Scripting](#).

Description

Return a list of Twilio accounts that have been configured in the Gateway as a single-column Dataset.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.twilio.getAccountsDataset()

- Parameters
 - Nothing
- Returns
 - Dataset - A list of configured Twilio accounts as a single-column dataset.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Retrieves a list of Twilio accounts and then passes the data to a Table component's data property.  
  
# Call system.twilio.getAccountsDataset() and store the returned list into a variable.  
twilioAccounts = system.twilio.getAccountsDataset()  
  
# Pass the dataset to a Table component. The Table is located in the same container as the  
# component calling this script.  
event.source.parent.getComponent('Table').data = twilioAccounts
```

Keywords

system twilio getAccountsDataset, twilio.getAccountsDataset

system.twilio.getPhoneNumbers

This function is used in [Python Scripting](#).

Description

Returns a list of outgoing phone numbers for a Twilio account. Note that these numbers are supplied by Twilio and are not defined on a user in Ignition.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.twilio.getPhoneNumbers(accountName)

- Parameters

[String](#) accountName - The Twilio account for which to retrieve phone numbers.

- Returns

[List](#) - A list of phone numbers for the given Twilio account.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Retrieves a list of phone numbers associated with a Twilio account and then iterates through the
# resulting list.
# Checks against a Twilio profile configured on the Gateway by the name of "Twilio Account".

# Call system.twilio.getPhoneNumbers() and store the returned list into a variable.
twilioNumbers = system.twilio.getPhoneNumbers("Twilio Account")

# Iterate through the list of numbers.
for number in twilioNumbers:

    # Prints the numbers to the console, but could do something more useful with each number.
    print number
```

Keywords

system twilio getPhoneNumbers, twilio.getPhoneNumbers

system.twilio.getPhoneNumbersDataset

This function is used in [Python Scripting](#).

Description

Return a list of outgoing phone numbers for a Twilio account as a single-column Dataset. Note that these numbers are supplied by Twilio and are not defined on a user in Ignition.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.twilio.getPhoneNumbersDataset(accountName)

- Parameters

[String](#) accountName - The Twilio account for which to retrieve phone numbers.

- Returns

[Dataset](#) - A list of phone numbers for the given Twilio account as a single-column dataset,

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Retrieves a list of phone numbers associated with a Twilio account and then passes the resulting list to a Table component's Data property.  
# Checks against a Twilio profile configured on the Gateway by the name of "Twilio Account".  
  
# Call system.twilio.getPhoneNumbers() and store the returned list into a variable.  
twilioNumbers = system.twilio.getPhoneNumbersDataset("Twilio Account")  
  
# Pass the dataset to a Table component. The Table is located in the same container as the  
# component calling this script.  
event.source.parent.getComponent('Table').data = twilioNumbers
```

Keywords

system twilio getPhoneNumbersDataset, twilio.getPhoneNumbersDataset

system.twilio.sendSms

This function is used in [Python Scripting](#).

Description

Sends an SMS message.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.twilio.sendSms(accountName, fromNumber, toNumber, message)

- Parameters

String accountName - The Twilio account to send the SMS from.

String fromNumber - The outbound phone number belonging to the Twilio account to use.

String toNumber - The phone number of the recipient.

String message - The body of the SMS.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Send an SMS message.  
# Fetch the Twilio account name.  
# getAccounts() returns a list, so the "[0]" operator is referring to the first item in the list.  
account = system.twilio.getAccounts()[0]  
  
# Fetch the first number associated with the account.  
fromNumber = system.twilio.getPhoneNumbers(account)[0]  
  
# Fetch a specific user's contact information.  
# A static value is used below, but system.user.getUser() could be used to retrieved a user's phone  
number.  
toNumber = "+19165550101"  
  
# Define the text message.  
# A static message is used below, but multiple messages could be stored in a database table and retrieved  
here.  
textMessage = "This is the body of a text message"  
  
# Send the message.  
system.twilio.sendSms(account, fromNumber, toNumber, textMessage)
```

Keywords

system twilio.sendSms, twilio.sendSms

system.user

User Functions

The following functions give you access to view (and edit, if the user source supports editing) users in any user source.

[In This Section ...](#)

system.user.addCompositeSchedule

This function is used in **Python Scripting**.

Description

Allows two schedules to be combined into a composite schedule.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.addCompositeSchedule(name, scheduleOne, scheduleTwo, [description])

- Parameters

- string name - The name of the new composite schedule.
 - string scheduleOne - The first schedule to combine.
 - string scheduleTwo - The second schedule to combine.
 - string description - Description of the new combined schedule. [optional]

- Returns

- UIResponse - A [UIResponse](#) object with lists of warnings, errors, and info about the success or failure of the add.

- Scope

- Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Creating a Composite Schedule

```
# Assuming you already have two schedules configured, named "A Shift" and "B Shift",
# you could create a composite schedule with the following.
system.user.addCompositeSchedule("A and B Shift", "A Shift", "B Shift", "Both A and B combined")
```

Keywords

system user addCompositeSchedule, user.addCompositeSchedule

system.user.addHoliday

This function is used in **Python Scripting**.

Description

Allows a holiday to be added.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.addHoliday(holiday)

- Parameters

[HolidayModel](#) **holiday** - The holiday to add, as a [HolidayModel](#) object.

- Returns

[UIResponse](#) - A [UIResponse](#) object with lists of warning, errors and info about the success or failure of the add.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Adding a Holiday

```
# This example adds a holiday.
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

from com.inductiveautomation.ignition.common.user.schedule import HolidayModel
from java.util import Date
holidayName = "Groundhog Day"
d = Date(2016 - 1900, 2, 2)                                # java dates start in 1900
repeatAnnually = False
myHoliday = HolidayModel(holidayName, d, repeatAnnually)
response = system.user.addHoliday(myHoliday)

warnings = response.getWarns()
print "Warnings are:"
printResponse(warnings)

errors = response.getErrors()
print "Errors are:"
printResponse(errors)

infos = response.getInfo()
print "Infos are:"
printResponse(infos)
```

```
"""The example above outputs the following:  
Warnings are:  
None  
Errors are:  
None  
Infos are:  
New holiday "Groundhog Day" added.  
"""
```

Keywords

system user addHoliday, user.addHoliday

system.user.addRole

This function is used in **Python Scripting**.

Description

Adds a role to the specified user source. When altering the Gateway system user source, the [Allow User Admin](#) setting must be enabled.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.addRole(userSource, role)

- Parameters

 String userSource - The user source to add a role to. Blank will use the default user source.

 String role - The role to add. Role must not be blank and must not already exist.

- Returns

 UIResponse - A [UIResponse](#) object with lists of warnings, errors, and info about the success or failure of the add.

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example adds a role "Operator" to the user source "MyUserSource".  
system.user.addRole("MyUserSource", "Operator")
```

Keywords

system user addRole, user.addRole

system.user.addSchedule

This function is used in **Python Scripting**.

Description

Adds a schedule.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.addSchedule(schedule)

- Parameters

ScheduleModel *schedule* - The schedule to add. Can be a **BasicScheduleModel** or **CompositeScheduleModel** object (or any other class that extends **AbstractScheduleModel**).

- Returns

UIResponse - A **UIResponse** object with lists of warnings, errors, and info about the success or failure of the add.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Adding Schedule

```
# This example tries to add the schedule NewSchedule based on an existing schedule MySchedule, and prints the results of the action.

# This function prints the response received.
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

# The main function.
mySchedule = system.user.getSchedule("Always")
if mySchedule != None and mySchedule.getType() == "basic schedule":
    mySchedule.setObserveHolidays(False)
    mySchedule.setName("NewSchedule")
    response = system.user.addSchedule(mySchedule)
    warnings = response.getWarns()
    print "Warnings are:"
    printResponse(warnings)

    errors = response.getErrors()
    print "Errors are:"
    printResponse(errors)

    infos = response.getInfo()
    print "Infos are:"
    printResponse(infos)
```

```
"""The example above outputs the following:  
Warnings are:  
None  
Errors are:  
None  
Infos are:  
New schedule "NewSchedule" added.  
"""
```

Keywords

system user addSchedule, user.addSchedule

system.user.addUser

This function is used in **Python Scripting**.

Description

Adds a new user to a user source. Used in combination with [getNewUser](#) to create new user.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.addUser(userSource, user)

- Parameters

String userSource - The user source to add a user to. If set to an empty string, the function will attempt to use the project's default user source (if called from a project).

User user - The user to add, as a [User](#) object. Refer also to the [PyUser](#) class.

- Returns

UIResponse - A [UIResponse](#) object which contains lists of the errors, warnings, and information returned after the add attempt.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Adding a New User

```
# Get new user.
userToGet = system.user.getNewUser("AcmeWest", "mTrejo")

# Add some contact info.
contactInfo = {"email": "mTrejo@acmewest.com", "sms": "5551234"}
userToGet.addContactInfo(contactInfo)
userToGet.set("password", "thisIsMyPassword")

# Adds a user to the AcmeWest usersource.
system.user.addUser("AcmeWest", userToGet)
```

Keywords

system user addUser, user.addUser

system.user.createScheduleAdjustment

This function is used in **Python Scripting**.

Description

Creates a schedule adjustment.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.createScheduleAdjustment(startDate, endDate, isAvailable, note)

- Parameters

Date startDate - The starting date of the schedule adjustment.

Date endDate - The ending date of the schedule adjustment.

Boolean isAvailable - True if the user is available during this schedule adjustment.

String note - A note about the schedule adjustment.

- Returns

Schedule Adjustment - A **ScheduleAdjustment** object that can be added to a user.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Creating Schedule Adjustment

```
# Specify the range of the schedule change.  
start = system.date.parse("2019-07-01 17:00:00")  
end = system.date.parse("2019-07-05 17:00:00")  
  
# Create an adjusted schedule.  
scheduleAdjustment = system.user.createScheduleAdjustment(start, end, True, "Summer swing schedule  
change.")  
  
# Get the user we need to adjust.  
user = system.user.getUser("default", "george")  
  
# Apply the adjusted schedule to the temporary user that lives in this script.  
user.addScheduleAdjustments([scheduleAdjustment])  
  
# Override the old george user in the user source, with the new user we created in this script.  
system.user.editUser("default", user)
```

Keywords

system user createScheduleAdjustment, user.createScheduleAdjustment

system.user.editHoliday

This function is used in **Python Scripting**.

Description

Allows a holiday to be edited.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.editHoliday(holidayName, holiday)

- Parameters

String **holidayName** - The name of the holiday to edit. Name is case-sensitive.

HolidayModel **holiday** - The edited holiday, as a **HolidayModel** object.

- Returns

UIResponse - A **UIResponse** object with lists of warnings, errors, and info about the success or failure of the edit.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example gets a holiday and edits it.

# This function prints the response received.
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

# The main function.
holidayName = "Labor Day"
myHoliday = system.user.getHoliday(holidayName)
if myHoliday != None:
    myHoliday.setRepeatAnnually(False)
    response = system.user.editHoliday(holidayName, myHoliday)

    warnings = response.getWarns()
    print "Warnings are:"
    printResponse(warnings)

    errors = response.getErrors()
    print "Errors are:"
    printResponse(errors)

    infos = response.getInfo()
    print "Infos are:"
```

```
printResponse(info)

"""The example above outputs the following:
Warnings are:
None
Errors are:
None
Infos are:
Holiday "Labor Day" updated.
"""


```

Keywords

system user editHoliday, user.editHoliday

system.user.editRole

This function is used in **Python Scripting**.

Description

Renames a role in the specified user source. When altering the Gateway System User Source, the [Allow User Admin](#) setting must be enabled.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.editRole(userSource, oldName, newName)

- Parameters

 String userSource - The user source in which the role is found. Blank will use the default user source.

 String oldName - The role to edit. Role must not be blank and must exist.

 String newName - The new name for the role. Must not be blank.

- Returns

 UIResponse - A [UIResponse](#) object with lists of warnings, errors, and info about the success or failure of the edit.

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example edits the role "Operator" in the user source "MyUserSource" and edits it to the role "User".  
system.user.editRole("MyUserSource", "Operator", "User")
```

Keywords

system user editRole, user.editRole

system.user.editSchedule

This function is used in **Python Scripting**.

Description

Allows a schedule to be edited.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.editSchedule(scheduleName, schedule)

- Parameters

String scheduleName - The name of the schedule to edit. Name is case-sensitive.

ScheduleModel schedule - The schedule to add. Can be a [BasicScheduleModel](#) or [CompositeScheduleModel](#) object (or any other class that extends [AbstractScheduleModel](#)).

- Returns

UIResponse - A [UIResponse](#) object with lists of warnings, errors, and info about the success or failure of the edit.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Editing Schedule

```
# This example tries to edit the schedule MySchedule, and prints the results of the action.

# This function prints the response received.
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

# The main function.
oldScheduleName = "MySchedule"
mySchedule = system.user.getSchedule(oldScheduleName)
if mySchedule != None and mySchedule.getType() == "basic schedule":
    mySchedule.setObserveHolidays(False)
    mySchedule.setName("MyEditedSchedule")
    mySchedule.setDescription("A modified description")
    response = system.user.editSchedule(oldScheduleName, mySchedule)
    warnings = response.getWarns()
    print "Warnings are:"
    printResponse(warnings)

    errors = response.getErrors()
    print "Errors are:"
    printResponse(errors)
```

```
infos = response.getInfos()
print "Infos are:"
printResponse(infos)
else:
    print "Basic schedule", oldScheduleName, "not found."
"""The example above outputs the following:Warnings are:
None
Errors are:
None
Infos are:
Schedule "MyEditedSchedule" updated."""
```

Keywords

system user editSchedule, user.editSchedule

system.user.editUser

This function is used in **Python Scripting**.

Description

Alters a specific user in a user source, replacing the previous data with the new data passed in.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.editUser(userSource, user)

- Parameters

String userSource - The user source in which the user is found. Blank will use the default user source.

User user - The user to update, as a [User](#) object. Refer also to the [PyUser](#) class.

- Returns

UIResponse - A [UIResponse](#) object with lists of warnings, errors, and information returned after the edit attempt.

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Editing a User

```
# Retrieve the user we're going to edit.  
userToChange = system.user.getUser("default", "george")  
  
# Make a change to the user. In this case, we're adding some contact info.  
contactInfo = {"email":"ignition_user@mycompany.com", "sms": "5551212"}  
userToChange.addContactInfo(contactInfo)  
  
# Edit the user. Because the user object we're passing in has a user name, the function  
# already knows which user to edit.  
system.user.editUser("default", userToChange)
```

Keywords

system user editUser, user.editUser

system.user.getHoliday

This function is used in [Python Scripting](#).

Description

Returns a specific holiday.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getHoliday(holidayName)

- Parameters

[String](#) `holidayName` - The name of the holiday to return. Case-sensitive.

- Returns

[HolidayModel](#) - The holiday, as a [HolidayModel](#) object, or None if not found.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will get a holiday and print info about it.
holidayName = "Labor Day"
holiday = system.user.getHoliday(holidayName)
if holiday == None:
    print holidayName, "not found"
else:
    print holiday.getName(), holiday.getDate(), holiday.isRepeatAnnually()

"""The example above outputs the following:
Labor Day 2015-09-07 00:00:00.0 False
"""


```

Keywords

system user getHoliday, user.getHoliday

system.user.getHolidayNames

This function is used in [Python Scripting](#).

Description

Returns a collection of strings of all holiday names.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getHolidayNames()

- Parameters
 - Nothing
- Returns
 - [List](#) - A list of all holiday names, or an empty list if no holidays are defined.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example prints the name of every holiday.

holidayNames = system.user.getHolidayNames()
for holidayName in holidayNames:
    print holidayName
```

Keywords

system user getHolidayNames, user.getHolidayNames

system.user.getHolidays

This function is used in [Python Scripting](#).

Description

Returns a sequence of all of the holidays available.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getHolidays()

- Parameters
 - Nothing
- Returns
 - [List](#) - A list of holidays, as [HolidayModel](#) objects.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example prints information about every holiday.
holidays = system.user.getHolidays()
if len(holidays) == 0:
    print "No holidays defined"
for holiday in holidays:
    print holiday.getName(), holiday.getDate(), holiday.isRepeatAnnually()
```

Keywords

system user getHolidays, user.getHolidays

system.user.getNewUser

This function is used in **Python Scripting**.

Description

Creates a new user object. The user will not be added to the user source until [addUser](#) is called.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.getNewUser(userSource, username)

- Parameters

String userSource - The name of the user source in which to create a user.

String username - The username for the new user. Does not check if username already exists or is valid.

- Returns

User - The new user, as a [User](#) object. Refer also to the [PyUser](#) class.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Get new user.
userToGet = system.user.getNewUser("AcmeWest", "mTrejo")

# Add some contact info.
contactInfo = {"email": "mTrejo@acmewest.com", "sms": "5551234"}
userToGet.addContactInfo(contactInfo)
userToGet.set("password", "mypassword")

# Adds a user to the the AcmeWest usersource.
system.user.addUser("AcmeWest", userToGet)
```

Code Snippet

```
# Util for printing the responses.
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

# Make a brand new 'blank' user. Not saved until we, well, save.
username = event.source.parent.getComponent('Text Field').text
```

```

user = system.user.getNewUser("", "myAwesomeUser")

# Let's fill in some fields. Note we have two ways to access property names.
user.set("firstname", "Naomi")
user.set(user.LastName, "Nagata")
user.set("password", "1234567890")

# We can add contact info one at a time. Up to the script user to make sure the type is legit.
user.addContactInfo("email", "naomi@roci.com")

# We can add a lot of contact info.
contactInfo = {"email": "ignition_user@mycompany.com", "sms": "5551212"}
user.addContactInfo(contactInfo)

# We can delete contact info. Only deletes if both fields match.
user.removeContactInfo("sms", "5551212")

# We can add a role. If the role doesn't already exist, user save will fail, depending on user source.
user.addRole("mechanic")

# We can add a lot of roles.
roles = ["Administrator", "prisoner"]
user.addRoles(roles)

# We can remove a role.
user.removeRole("prisoner")

# We can add a schedule adjustment too.
date2 = system.date.now()
date1 = system.date.midnight(date2)
user.addScheduleAdjustment(date1, date2, False, "An adjustment note")

# We can make a bunch of adjustments and add them en-masse.
date3 = system.date.addDays(date2, -4)
adj1 = system.user.createScheduleAdjustment(date3, date2, True, "Another note")
adj2 = system.user.createScheduleAdjustment(date3, date1, False, "")
user.addScheduleAdjustments([adj1, adj2])

# and we can remove a schedule adjustment. All fields must match.
user.removeScheduleAdjustment(date1, date2, True, "Some other note")

# Finally need to save our new user and print responses.
response = system.user.addUser("", user)

warnings = response.getWarns()
print "Warnings are:"
printResponse(warnings)

errors = response.getErrors()
print "Errors are:"
printResponse(errors)

infos = response.getInfo()
print "Infos are:"
printResponse(infos)

```

Keywords

system user getNewUser, user.getNewUser

system.user.getRoles

This function is used in [Python Scripting](#).

Description

Returns a sequence of strings representing all of the roles configured in a specific user source.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getRoles(userSource)

- Parameters

[String](#) userSource - The user source to fetch the roles for.

- Returns

[List](#) - A List of strings that holds all the roles in the user source.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Get All Roles

```
# This example will print a list of all user roles in the default user source.  
  
roles = system.user.getRoles("")  
for role in roles:  
    print role
```

Keywords

system user getRoles, user.getRoles

system.user.getSchedule

This function is used in [Python Scripting](#).

Description

Returns a specific schedule.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getSchedule(scheduleNames)

- Parameters

[String](#) scheduleName - The name of the schedule to return. Case-sensitive.

- Returns

[ScheduleModel](#) - The schedule, which can be a [BasicScheduleModel](#) object, [CompositeScheduleModel](#) object, or another type registered by a module. If a schedule was not found, the function will return None if called from a Vision Client or the Designer. If called in from a Perspective Session or anywhere else in the Gateway scope, will throw an [IllegalArgumentException](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Showing Schedule Information

```
# This example will get a schedule and print info about it.

# This function handles recursive printing of the different schedule types. Modules can register more
types than listed here.
def printScheduleInfo(aSchedule):
    if aSchedule.getType() == "basic schedule":
        print "Basic schedule type: ",aSchedule.getName(), aSchedule.getDescription(), aSchedule.
isAllDays(), aSchedule.isObserveHolidays()
    elif aSchedule.getType() == "composite schedule":
        compositePieces = aSchedule.getModels()
        print "Composite schedule type:",aSchedule.getName(), aSchedule.getDescription(), " which
is made up of..."
        for piece in compositePieces:
            printScheduleInfo(piece)
    else:
        print "Other schedule type: ", aSchedule.getName(), aSchedule.getDescription(), aSchedule.
getType(), aSchedule.isObserveHolidays()

# The main function.
scheduleName = "MySchedule"
schedule = system.user.getSchedule(scheduleName)
if schedule == None:
    print "Schedule", scheduleName, "was not found"
else:
    printScheduleInfo(schedule)

"""The example above outputs the following:
Basic schedule type: MySchedule A description False True"""
```

Keywords

system user getSchedule, user.getSchedule

system.user.getScheduledUsers

This function is used in [Python Scripting](#).

Description

Returns a list of users that are scheduled on. If no users are scheduled, it will return an empty list.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getScheduledUsers(userSource, [date])

- Parameters

[String](#) userSource - The name of the user source to check for scheduled users.

[Date](#) date - The date to check schedules for. May be a Java Date or Unix Time in ms. If omitted, the current date and time will be used. [optional]

- Returns

[List](#) - List of all Users (as [User](#) objects) scheduled for the given date, taking schedule adjustments into account. Refer also to the [PyUser](#) class.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Get all users scheduled for the date specified in a popup calendar and print their names.

date = event.source.parent.getComponent('Popup Calendar').date
users = system.user.getScheduledUsers("default", date)

if users == None:
    print "No users scheduled"
else:
    print "Scheduled users:"
    for user in users:
        print user.get(user.Username)
```

Keywords

system user getScheduledUsers, user.getScheduledUsers

system.user.getScheduleNames

This function is used in [Python Scripting](#).

Description

Returns a sequence of strings representing the names of all of the schedules available.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getScheduleNames()

- Parameters
 - Nothing
- Returns
 - [List](#) - A List of Strings that holds the names of all the available schedules.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print a list of all available schedules.  
  
schedules = system.user.getScheduleNames()  
for schedule in schedules:  
    print schedule
```

Keywords

system user getScheduleNames, user.getScheduleNames

system.user.getSchedules

This function is used in **Python Scripting**.

Description

Returns a sequence of all available schedule models, which can be used to return configuration information on the schedule, such as time for each day of the week.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getSchedules()

- Parameters

Nothing

- Returns

[List](#) - A list of schedules. Each schedule can be a [BasicScheduleModel](#) object, [CompositeScheduleModel](#) object, or another type registered by a module.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print a list of all available ScheduleModels.

# This function handles recursive printing of the different schedule models. Modules can register more
types than listed here.
def printScheduleInfo(aSchedule):
    if aSchedule.getType() == "basic schedule":
        print "Basic schedule type: ",aSchedule.getName(), aSchedule.getDescription(), aSchedule.
isAllDays(), aSchedule.getAllDayTime()
    elif aSchedule.getType() == "composite schedule":
        compositePieces = aSchedule.getModels()
        print "Composite schedule type:",aSchedule.getName(), aSchedule.getDescription(), " which
is made up of..."
        for piece in compositePieces:
            printScheduleInfo(piece)
    else:
        print "Other schedule type: ", aSchedule.getName(), aSchedule.getDescription(), aSchedule.
getType(), aSchedule.isObserveHolidays()

# The main function.
schedules = system.user.getSchedules()
for schedule in schedules:
    printScheduleInfo(schedule)
```

Keywords

system user.getSchedules, user.getchedules

system.user.getUser

This function is used in [Python Scripting](#).

Description

Looks up a specific user in a user source, by username. The full User object is returned except for the user's password.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getUser(userSource, username)

- Parameters

[String](#) userSource - The name of the user source to search for the user in. Can be a blank string to use the Vision Client's default user source.

[String](#) username - The username of the user to search for.

- Returns

[User](#) - The user, as a [User](#) object. Refer also to the [PyUser](#) class.

- Scope

Gateway, Vision Client, Perspective Session

User Object

The "User" object that is returned contains all of the information about that user, except for the user's password. You can access most of the basic user properties via a call to "get" or "getOrDefault" which returns a default value if the requested item is not present. For example:

```
user.getOrDefault('schedule')
```

This will return that user's schedule, or the value of "Always" if no schedule has been set as that is the default schedule. The following are the various values you may use in this manner:

- username
- firstname
- lastname
- notes
- schedule
- language

In addition to these properties, the user object has other methods on it to retrieve more information:

- [user.getId\(\)](#) - Returns the internal identifier object that the backing user source needs to identify this user.
- [user.getRoles\(\)](#) - Returns a sequence of strings representing the roles that this user belongs to.
- [user.getContactInfo\(\)](#) - Returns a sequence of ContactInfo objects. Each of these objects will have a contactType and a value property representing the contact information. Both properties are strings.
- [user.getScheduleAdjustments\(\)](#) - Returns a sequence of ScheduleAdjustment objects. Each of these objects will have two date properties, "start" and "end", a boolean property, "available", and a string property called "note".
- [user.getPath\(\)](#) - Returns a QualifiedPath object that represents this user in a deterministic manner.

Code Examples

Code Snippet

```
# This example will print the first and last name of the current user using the default datasource.  
userName = system.security.getUsername()  
user = system.user.getUser("", userName)  
print user.get('firstname') + " " + user.get('lastname')
```

Keywords

system user getUser, user.getUser

system.user.getUsers

This function is used in **Python Scripting**.

Description

Retrieves the list of users in a specific user source. The "User" objects that are returned contain all of the information about that user, except for the user's password.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.getUsers(userSource)

- Parameters

String userSource - The name of the user source to find the users in.

- Returns

List - A list of [User](#) objects. Refer also to the [PyUser](#) class.

- Scope

Gateway, Vision Client, Perspective Session

User Object

The "User" object that is returned contains all of the information about that user, except for the user's password. You can access most of the basic user properties via a call to "get" or "getOrDefault" which returns a default value if the requested item is not present. For example:

```
user.getOrDefault('schedule')
```

This will return that user's schedule, or the value of "Always" if no schedule has been set as that is the default schedule. The following are the various values you may use in this manner:

- username
- firstname
- lastname
- notes
- schedule
- language

In addition to these properties, the user object has other methods on it to retrieve more information:

- **user.getId()** - Returns the internal identifier object that the backing user source needs to identify this user.
- **user.getRoles()** - Returns a sequence of strings representing the roles that this user belongs to.
- **user.getContactInfo()** - Returns a sequence of ContactInfo objects. Each of these objects will have a contactType and value property representing the contact information, both strings.
- **user.getScheduleAdjustments()** - Returns a sequence of ScheduleAdjustment objects. Each of these objects will have two date properties, "start" and "end", a boolean property, "available", and a string property called "note".
- **user.getPath()** - Returns a QualifiedPath object that represents this user in a deterministic manner.

Code Examples

Code Snippet

```
# This example will print the first and last name of all users, using the default datasource.
```

```
users = system.user.getUsers("")  
for user in users:  
    print user.get('firstname') + " " + user.get('lastname')
```

Keywords

system user getUsers, user.getUsers

system.user.isUserScheduled

This function is used in [Python Scripting](#).

Description

Will check if a specified user is scheduled currently or on a specified date/time.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.user.isUserScheduled(user, [date])

- Parameters

[User](#) user - The [User](#) object to check for on the schedule.

[Date | Integer](#) date - The date to check schedules for. May be a Java Date or Unix Time in ms. If omitted, the current date and time will be used. [optional]

- Returns

[Boolean](#) - True if the user is scheduled for the specified date; false if not.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Print whether or not the user "oper1" is scheduled currently.

user = system.user.getUser("", "oper1")
if system.user.isUserScheduled(user):
    print "oper1 is scheduled"
else:
    print "oper1 is not scheduled"
```

Keywords

system user isUserScheduled, user.isUserScheduled

system.user.removeHoliday

This function is used in **Python Scripting**.

Description

Allows a holiday to be deleted.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.removeHoliday(holidayName)

- Parameters

String **holidayName** - The name of the holiday to delete. Case-sensitive.

- Returns

UIResponse - A list of **UIResponse** objects with lists of warnings, errors, and info about the success or failure of the deletion.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

holidayName = "Labor Day"
response = system.user.removeHoliday(holidayName)

warnings = response.getWarns()
print "Warnings are:"
printResponse(warnings)

errors = response.getErrors()
print "Errors are:"
printResponse(errors)

infos = response.getInfo()
print "Infos are:"
printResponse(infos)

"""The example above outputs the following:
Warnings are:
None
Errors are:
None
Infos are:
```

```
Holiday "Labor Day" deleted.  
"""
```

Keywords

system user removeHoliday, user.removeHoliday

system.user.removeRole

This function is used in **Python Scripting**.

Description

Removes a role from the specified user source. When altering the Gateway System User Source, the [Allow User Admin](#) setting must be enabled.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.removeRole(userSource, role)

- Parameters

 String userSource - The user source in which the role is found. Blank will use the default user source.

 String role - The role to remove. The role must exist.

- Returns

 UIResponse - A list of [UIResponse](#) objects with lists of warnings, errors, and info about the success or failure of the deletion

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Removes the role "User" in the user source "MyUserSource".  
system.user.removeRole("MyUserSource", "User")
```

Keywords

system user removeRole, user.removeRole

system.user.removeSchedule

This function is used in **Python Scripting**.

Description

Allows a schedule to be deleted. Note that schedules which are used in Composite Schedules can not be deleted until they are removed from the Composite Schedule.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.removeSchedule(scheduleName)

- Parameters

String `scheduleName` - The name of the schedule to delete. Case-sensitive.

- Returns

UIResponse - A list of `UIResponse` objects with lists of warnings, errors, and info about the success or failure of the deletion.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example tries to delete the schedule MySchedule, and prints the results of the action.
```

```
def printResponse(responseList):
    if len(responseList) > 0:
        for response in responseList:
            print "", response
    else:
        print " None"

scheduleName = "MySchedule"
response = system.user.removeSchedule(scheduleName)

warnings = response.getWarns()
print "Warnings are:"
printResponse(warnings)

errors = response.getErrors()
print "Errors are:"
printResponse(errors)

infos = response.getInfos()
print "Infos are:"
printResponse(infos)

"""The example above outputs the following:
Warnings are:
None
Errors are:
```

```
None  
Infos are:  
Schedule "MySchedule" deleted.  
"""
```

Keywords

system user removeSchedule, user.removeSchedule

system.user.removeUser

This function is used in **Python Scripting**.

Description

Removes a specific user from the a user source based on username. When altering the Gateway System User Source, the [Allow User Admin](#) setting must be enabled.

Client Permission Restrictions

Permission Type: User Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.user.removeUser(userSource,username)

- Parameters

String userSource - The user source in which the user is found. Blank will use the default user source.

String username - The username of the user to remove.

- Returns

UIResponse - An [UIResponse](#) object with lists of warnings, errors, and information returned after the removal attempt.

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Removes user jHopkins from the AcmeWest user source.  
system.user.removeUser("AcmeWest", "jHopkins")
```

Keywords

system user removeUser, user.removeUser

system.util

Utility Functions

The following functions give you access to view various Gateway and Client data, as well as interact with other various systems.

[In This Section ...](#)

system.util.audit

This function is used in [Python Scripting](#).

Description

Inserts a record into an audit profile.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.util.audit([action], [actionValue], [auditProfile], [actor], [actorHost], [originatingSystem], [eventTimestamp], [originatingContext], [statusCode])`

- Parameters

`String` `action` - What happened. Default is null. [optional]

`String` `actionTarget` - What the action happened to. Default is null. [optional]

`String` `auditProfile` - Where the audit record should be stored. Defaults to the project's audit profile (if specified), or the Gateway audit profile if calling in the Gateway or Perspective Session scope. [optional]

`String` `actor` - Who made the change. Will be populated automatically if omitted, assuming there is a known user. [optional]

`String` `actorHost` - The hostname of whoever made the change. Will be populated automatically if omitted. [optional]

`List[String], String | String` `originatingSystem` - An even-length list providing additional context to the audit event. Will be appended to the automatically generated list. Typically, the automatically generated context looks like this: `sys:${gatewayName}:\project:${projectId}`. So if you provided `["component", "Joe'sButton", "field", "value"]`, you would get a record with `originatingSystem:sys:${gatewayName}:\project:${projectId}:\component:Joe'sButton:field:value`. Or, if a string is provided, this automatic context will not be used and your provided string will be written directly into the `originatingSystem` column in the audit profile. [optional]

`Date` `eventTimestamp` - When the event happened. Will be set to the current time if omitted. [optional]

`Integer` `originatingContext` - What scope the event originated from: 1 means Gateway, 2 means Designer, 4 means Client. Will be set automatically if omitted. [optional]

`Integer` `statusCode` - A quality code to attach to the object. Defaults to 0, indicating no special meaning. [optional]

- Scope

Gateway, Vision Client, and Perspective Session.

Code Examples

```
# All of the parameters are optional, so you're free to only provide parameters you're interested in.  
# In the very least provide just the action you wish to record, which gives the function a chance to look  
up all of the other parameters automatically.  
system.util.audit("The user did a thing!")
```

```
# Simple example just showing parameter usage.  
myAction = "The button was pressed"  
myTarget = "My Button"  
  
system.util.audit(action = myAction , actionTarget = myTarget )
```

Keywords

system util audit, util.audit

system.util.beep

This function is used in [Python Scripting](#).

Description

Tells the computer where the script is running to make a "beep" sound. The computer must have a way of producing sound.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.beep()

- Parameters

Nothing

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This will simply cause the system where the script is being executed to emit a beep sound.  
# That system must have a way to produce sound, such as speakers or headphones.
```

```
system.util.beep()
```

Keywords

system util beep, util.beep

system.util.execute

This function is used in [Python Scripting](#).

Description

Executes the given commands via the operating system, in a separate process. The commands argument is an array of strings. The first string is the program to execute, with subsequent strings being the arguments to that command.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.execute(commands)

- Parameters

[List\[String\]](#) commands - A list containing the command (1st entry) and associated arguments (remaining entries) to execute.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code starts the Firefox browser and opens the Google home page.  
system.util.execute(['C:\\Program Files\\Mozilla Firefox\\firefox.exe', 'https://www.google.com'])
```

Code Snippet

```
# This function flushes the resolver cache.  
system.util.execute(["ipconfig", "/flushdns"])
```

Code Snippet

```
# This code runs the Notepad program and opens the Ignition license text file.  
system.util.execute(['notepad.exe', 'C:\\Program Files\\Inductive Automation\\i\\Ignition\\license.txt'])
```

Keywords

system util execute, util.execute

system.util.exit

This function is used in [Python Scripting](#).

Description

Exits the running client, as long as the shutdown intercept script doesn't cancel the shutdown event. Set force to true to not give the shutdown intercept script a chance to cancel the exit. Note that this will quit the Client completely. you can use `system.security.logout()` to return to the login screen.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.exit([force])

- Parameters
 - `Boolean` force - If true, the shutdown-intercept script will be skipped. Default is false. [optional]
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code would exit the client after confirming with the user.  
if system.gui.confirm("Are you sure you want to exit?"):  
    system.util.exit()
```

Keywords

system util exit, util.exit

system.util.getAvailableLocales

This function is used in [Python Scripting](#).

Description

Returns a collection of strings representing the Locales added to the Translation Manager, such as 'en' for English.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getAvailableLocales()

- Parameters
 - Nothing
- Returns
 - [List](#) - A list of Java [Locale](#) objects.
- Scope
 - Vision Client

Code Examples

```
#This code will take all of the available locales, and put them into a text field on the same window.

collection = system.util.getAvailableLocales()
locales = ''
for locale in collection:
    if locales == '':
        locales += locale
    else:
        locales += ", " + locale
event.source.parent.getComponent('Text Field').text = locales
```

Keywords

[system util getAvailableLocales](#), [util.getAvailableLocales](#)

system.util.getAvailableTerms

This function is used in [Python Scripting](#).

Description

Returns a collection of available terms defined in the translation system.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getAvailableTerms()

- Parameters

Nothing

- Returns

[List](#) - A list of all of the terms available from the translation manager, as strings.

- Scope

Vision Client

Code Examples

```
# This code will print out a list of all of the available terms to the console.  
  
collection = system.util.getAvailableTerms()  
for term in collection:  
    print term
```

Keywords

system util getAvailableTerms, util.getAvailableTerms

system.util.getClientId

This function is used in [Python Scripting](#).

Description

Returns a hex-string that represents a number unique to the running client's session. You are guaranteed that this number is unique between all running clients.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getClientId()

- Parameters
 - Nothing
- Returns
 - [String](#) - A special code representing the client's session in a unique way.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code prints the current client's id to the debug console.  
id = system.util.getClientId()  
print id
```

Keywords

system util getClientId, util.ClientId

system.util.getConnectionMode

This function is used in [Python Scripting](#).

Description

Retrieves this Client's current connection mode: 3 is read/write, 2 is read-only, and 1 is disconnected.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getConnectionMode()

- Parameters
 - Nothing
- Returns
 - Integer** - The current connection mode for the Client.
- Scope
 - Vision Client

Code Examples

```
# This code sets a Client to read-only after a timeout of 30 seconds.  
# Add this code to a Client timer script.  
  
mode = system.util.getConnectionMode()  
if mode == 3 and system.util.getInactivitySeconds() > 30:  
    system.util.setConnectionMode(2)
```

Keywords

system util getConnectionMode, util.getConnectionMode

system.util.getConnectTimeout

This function is used in [Python Scripting](#).

Description

Returns the connect timeout in milliseconds for all Client-to-Gateway communication. This is the maximum amount of time that communication operations to the Gateway will be given to connect. The default is 10,000ms (10 seconds).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getConnectTimeout()

- Parameters
 - Nothing
- Returns
 - [Integer](#) - The current connect timeout, in milliseconds. Default is 10,000 (ten seconds).
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code prints out the current connect timeout.  
print system.util.getConnectTimeout()
```

Keywords

system util getConnectTimeout, util.getConnectTimeout

system.util.getEdition

This function is used in [Python Scripting](#).

Description

Returns the "edition" of the Vision Client: "standard", "limited", or "panel".

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getEdition()

- Parameters
 - Nothing
- Returns
 - String** - The edition of the Vision module that is running the Client.
- Scope
 - Vision Client

Code Examples

```
# This code writes the Vision module edition to a text field on the same page.  
event.source.parent.getComponent('Text Field').text = system.util.getEdition()
```

Keywords

system util getEdition, util.getEdition

system.util.getGatewayAddress

This function is used in [Python Scripting](#).

Description

Returns the address of the Gateway with which the Client is currently communicating.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getGatewayAddress()

- Parameters
 - Nothing
- Returns
 - String** - The address of the Gateway with which the client is communicating.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code opens up the Gateway Config page.  
address = system.util.getGatewayAddress()  
system.net.openURL("%s/web/config/" % address)
```

Keywords

system util getGatewayAddress, util.getGatewayAddress

system.util.getGatewayStatus

This function is used in [Python Scripting](#).

Description

Returns a string that indicates the status of the Gateway. A status of RUNNING means that the Gateway is fully functional. Thrown exceptions return "ERROR" with the error message appended to the string. This function can be used to test all 7.7 and later Gateways. The function can also be used to test 7.6 (7.6.4 and later) and 7.5 (7.5.11 and later) Gateways. Attempting to test Gateways older than these versions will return errors.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getGatewayStatus(gatewayAddress, [connectTimeoutMillis], [socketTimeoutMillis])

- Parameters

String gatewayAddress - The Gateway address to ping, in the form of ADDR:PORT/main.

Integer connectTimeoutMillis - The maximum time in milliseconds to attempt to initially contact a Gateway. [optional]

Integer socketTimeoutMillis - The maximum time in milliseconds to wait for a response from a Gateway after initial connection has been established. [optional]

- Returns

String - A string that indicates the status of the Gateway. A status of RUNNING means that the Gateway is fully functional.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
def checkRemoteGateway():

    status = system.util.getGatewayStatus("10.20.6.253:8088/main")

    if status == "RUNNING":
        print "Central Gateway is available!"
    else:
        print "Error: " + status

# It's important to do this as an asynchronous operation, as the method
# may block for some time.
system.util.invokeAsynchronous(checkRemoteGateway)
```

Keywords

system util getGatewayStatus, util.getGatewayStatus

system.util.getGlobals

This function is used in **Python Scripting**.

Description

This method returns a dictionary that provides access to the legacy global namespace. As of version 7.7.0, most new scripts use the modern style of scoping, which makes the 'global' keyword act very differently. Most importantly, the modern scoping rules mean that variables declared as 'global' are only global within that one module. The system.util.getGlobals() method can be used to interact with older scripts that used the old meaning of the 'global' keyword.

This feature is new in Ignition version **8.1.0**
[Click here](#) to check out the other new features

The globals dictionary will now persist across the lifetime of the JVM, and it's now accessible at [system.util.globals](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getGlobals()

- Parameters
 - Nothing
- Returns
 - [Dictionary\[String, Any\]](#) - The global namespace, as a dictionary.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Read and print out global variable 'foo'.
print system.util.getGlobals()['foo']
```

Code Snippet

```
# Write value 'hello' to global variable 'foo'.
system.util.getGlobals()['foo'] = 'hello'
```

Keywords

system util getGlobals, util.getGlobals

system.util.getInactivitySeconds

This function is used in **Python Scripting**.

Description

Returns the number of seconds since any keyboard or mouse activity.

Note: This function will always return zero in the Designer.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getInactivitySeconds()

- Parameters
 - Nothing
- Returns
 - Integer** - The number of seconds the mouse and keyboard have been inactive for this client.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This code could run in a global timer script.  
# After a 5-minute timeout, navigate back to the home screen.  
if system.util.getInactivitySeconds()>300 and system.nav.getCurrentWindow() != "Home":  
    system.nav.swapTo("Home")
```

Keywords

system util getInactivitySeconds, util.getInactivitySeconds

system.utilgetLocale

This function is used in [Python Scripting](#).

Description

Returns the current string representing the user's Locale, such as 'en' for English.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getLocale()

- Parameters
 - Nothing
- Returns
 - String** - String representing the user's Locale, such as 'en' for English.
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# Print a test if they are using English.
locale = system.util.getLocale()
if locale == "en":
    print "Using English"
```

Keywords

system util getLocale, util.getLocale

system.util.getLogger

This function is used in **Python Scripting**.

Description

Returns a Logger object that can be used to log messages to the console. Each Logger has a name, which is typically structured hierarchically using periods, indicating where in the project the Logger is used. You can use any naming scheme you like, however a well-planned naming scheme makes finding log entries and setting log levels much easier. Loggers can be shared between scripts simply by giving them the same name. Six levels of logging are available:

- Fatal - A severe error that will cause termination of the script.
- Error - A runtime error or other unexpected condition.
- Warn - An undesired condition, but one that does not interfere with execution.
- Info - An event that should be noted on the console, but is not an error.
- Debug - Detailed information useful in debugging.
- Trace - Highly detailed information.

To view log messages from Gateway scripts, in the Gateway go to Status > Diagnostics > Logs. To view log messages from Client scripts, including scripts in components, in the Client go to Help > Diagnostics > Log Viewer, or in the Designer go to Tools > Console. The default logging level is info, meaning that all messages with level info or higher are logged, and messages with a level of debug or trace are discarded.

To change the logging level for a Logger in a Gateway script, go to Status > Diagnostics > Log. Click the **Settings**  icon. The new logging level will remain until it is changed or the Gateway is restarted.

To change the logging level in a Client script, go to Help > Diagnostics > Logging Levels. Logging levels can not be changed in the Designer. The following methods are available to a Logger:

- `Logger.fatal(String)` - Logs a message with level fatal.
- `Logger.fatalf(String, Args...)` - Logs a formatted message with level fatal, using Java's Formatter syntax.
- `Logger.error(String)` - Logs a message with level error.
- `Logger.errorf(String, Args...)` - Logs a formatted message with level error, using Java's Formatter syntax.
- `Logger.warn(String)` - Logs a message with level warn.
- `Logger.warnf(String, Args...)` - Logs a formatted message with level warn, using Java's Formatter syntax.
- `Logger.info(String)` - Logs a message with level info.
- `Logger.infof(String, Args...)` - Logs a formatted message with level info, using Java's Formatter syntax.
- `Logger.debug(String)` - Logs a message with level debug.
- `Logger.debugf(String, Args...)` - Logs a formatted message with level debug, using Java's Formatter syntax.
- `Logger.trace(String)` - Logs a message with level trace.
- `Logger.tracef(String, Args...)` - Logs a formatted message with level trace, using Java's Formatter syntax.
- `Logger.isTraceEnabled()` - Returns True if the current log level is at least trace.
- `Logger.isDebugEnabled()` - Returns True if the current log level is at least debug.
- `Logger.isInfoEnabled()` - Returns True if the current log level is at least info.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getLogger(name)

- Parameters
 - `String name` - The name of a logger to create.
- Returns
 - `LoggerEx` - A new [LoggerEx](#) object used to log informational and error messages.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would log a message with level info.  
logger = system.util.getLogger("myLogger")  
logger.info("Hello, world.")
```

Code Snippet - Python String Formatting syntax

```
# This code would log a formatted message with level info.  
who = 'Bob Jones'  
num = 5  
logger = system.util.getLogger("myLogger")  
logger.info("Machine started by %s, employee ID %d" % (who, num))
```

Code Snippet - Java's Formatter syntax

```
# This code logs a formatted message with level info. Similar to the "Python String Formatting syntax" example above but using Java's Formatter syntax.  
# Note the 'f' at the end of the method name.  
who = 'Bob Jones'  
num = 5  
logger = system.util.getLogger("myLogger")  
logger.infof("Machine started by %s, employee ID %d", who, num)
```

Code Snippet

```
# This code would check if the debug level is enabled for this logger before executing the remaining code. Although not needed for a simple log entry like # in this example, it can eliminate expensive function calls in a more complex log entry.  
logger = system.util.getLogger("myLogger")  
if logger.isDebugEnabled():  
    logger.debug("Hello, world!")
```

Keywords

system util getLogger, util.getLogger

system.util.getProjectName

This function is used in **Python Scripting**.

Description

Returns the name of the project that is currently being run. When run from the Gateway scope from a resource that originates from a singular project (reports, SFCs, etc.), will return that resources project.

When called from a scope that does not have an associated project (a Tag Event Script), the function will return the name of the Gateway scripting project. If a Gateway scripting project has not been configured, then returns an empty string.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getProjectName()

- Parameters
 - Nothing
- Returns
 - String** - The name of the currently running project.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code displays the name of the currently running project.  
system.gui.messageBox("You are running project: %s" % system.util.getProjectName())
```

Keywords

system util getProjectName, util.get projectName

system.util.getProperty

This function is used in [Python Scripting](#).

Description

Retrieves the value of a named system property. Some of the available properties are:

- file.separator - The system file separator character, for example, "/" (unix) or "\" (windows).
- line.separator - The system line separator string, for example, "\r\n" (carriage return, newline).
- os.arch - Operating system architecture, for example, "x86".
- os.name - Operating system name, for example, "Windows XP".
- os.version - Operating system version, for example, "5.1".
- user.home - User's home directory.
- user.name - User's account name.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getProperty(propertyName)

- Parameters
 - String propertyName - The name of the system property to get.
- Returns
 - String - The value for the named property.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This script stores the contents of the Text Area component in the users home directory.
homeDir = system.util.getProperty("user.home")
sep = system.util.getProperty("file.separator")
path = "%s%sm myfile.txt" %(homeDir, sep)
system.file.writeFile(path, event.source.parent.getComponent("Text Area").text)
```

Keywords

system util getProperty, util.getProperty

system.util.getReadTimeout

This function is used in [Python Scripting](#).

Description

Returns the read timeout in milliseconds for all Client-to-Gateway communication. This is the maximum amount of time allowed for a communication operation to complete. The default is 60,000ms (1 minute).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getReadTimeout()

- Parameters

Nothing

- Returns

[Integer](#) - The current read timeout, in milliseconds. Default is 60,000 (one minute).

- Scope

Vision Client

Code Examples

```
# This code will find the current read timeout and write it to a numeric text field on the same page.  
event.source.parent.getComponent('Numeric Text Field').intValue = system.util.getReadTimeout()
```

Keywords

system util getReadTimeout, util.getReadTimeout

system.util.getSessionInfo

This function is used in [Python Scripting](#).

Description

Returns a PyDataSet holding information about all of the open Designer sessions and Vision Clients. Optional regular-expression based filters can be provided to filter the username or the username and the project returned.

The PyDataSet returned has these columns:

- username (String)
- project (String)
- address (String)
- isDesigner (Boolean)
- clientId (String)
- creationTime (Date)

This function will not return all sessions across a cluster - only the cluster node that is being communicated with by the client who makes the call.

Note: This function accepts [keyword arguments](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getSessionInfo([usernameFilter], [projectFilter])

- Parameters

[String](#) `usernameFilter` - A regular-expression based filter string to restrict the list by username. [optional]

[String](#) `projectFilter` - A regular-expression based filter string to restrict the list by project. [optional]

- Returns

[PyDataset](#) - A dataset representing the Gateway's current sessions.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code gets the entire table of sessions and puts it in an adjacent table.  
table = event.source.parent.getComponent("Table")  
sessions = system.util.getSessionInfo()  
table.data = system.db.toDataSet(sessions)
```

Code Snippet

```
# This code counts the number of times a user named "billy" is logged in.  
sessions = system.util.getSessionInfo("billy")  
system.gui.messageBox("Billy has %d sessions" % len(sessions))
```

Code Snippet

```
# This code returns session info on all users starting with the letters "bi".  
sessions = system.util.getSessionInfo("bi.*")
```

Code Snippet

```
# This code uses a single character wildcard in the username.  
sessions = system.util.getSessionInfo("bi.ly")
```

Code Snippet

```
# This code returns session info on a user named "bill.smith".  
sessions = system.util.getSessionInfo("bill\smith")
```

Keywords

system util getSessionInfo, util.getSessionInfo

system.util.getSystemFlags

This function is used in **Python Scripting**.

Description

Returns an integer that represents a bit field containing information about the currently running system. Each bit corresponds to a specific flag as defined in the bitmask below. The integer return will be a total of all of the bits that are currently active. See the example for tips on how to extract the information in this bit field. Note that the tag[System]Client/System/SystemFlags contains the same value.

Flag	Flag Description	Bit Value
Designer Flag	Set if running in the Designer.	1
Preview Flag	Set if running in the Designer, and the Designer is in preview mode.	2
Client Flag	Set if running as a Client.	4
Webstart Flag	Set if running as a Client in Web Start mode.	8
Applet Flag	Set if running as a Client in Applet mode.	16
Fullscreen Flag	Set if running as a Client in full screen mode.	32
SSL Flag	Set if communication to the Gateway is encrypted with SSL.	64
Mobile Flag	Set if currently running a mobile-launched client.	128

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getSystemFlags()

- Parameters
 - Nothing
- Returns
 - Integer** - A total of all the bits that are currently active. A full-screen Client launched from the Gateway webpage with no SSL will have a value of 44 (Fullscreen flag + Webstart Flag + Client Flag).
- Scope
 - Vision Client

Code Examples

```
# The first part of the script takes the integer representing the system flags, converts it to bits,
# places it in a list, and then prints it out.
# The second part of the script takes the list of bits, and places it in a table showing what each of the
# bits represent.

myList = []
flags = system.util.getSystemFlags()
for i in range(7,-1,-1):
    myList.insert(0, flags >> i & 1)
print myList

headers = ["Designer Flag", "Preview Flag", "Client Flag", "Webstart Flag", "Applet Flag", "Fullscreen
Flag", "SSL Flag", "Mobile Flag"]
data = system.dataset.toDataSet(headers, [myList])
```

```
table = event.source.parent.getComponent("Table")
table.data = data
```

Keywords

system util getSystemFlags, util.getSystemFlags

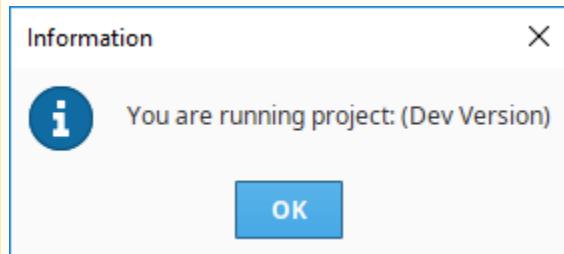
system.util.getVersion

This function is used in **Python Scripting**.

Description

Returns the Ignition version number that is currently being run.

Note: If the version is from a nightly build or developer version that is not yet released, the version number will come back as "Dev Version", for example:



Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.getVersion()

- Parameters
Nothing
- Returns
[Version](#) - The currently running Ignition version number, as a [Version](#) object.
- Scope
Gateway, Vision Client, Perspective Session

This section documents available attributes on the object.

Method and /or Attribute	Description	Return type
.major	Returns only the major version number. 8.0.2 returns 8	Integer
.minor	Returns only the minor version number. 8.0.2 returns 0	Integer
isFutureVersion()	Takes in a string version number and returns whether the current version is greater than the given version (true or false). Note: this does account for Snapshot, RC or Beta versions. Version format expected: "X.X.X" ie "8.0.7" See example below.	Boolean

Code Examples

Code Snippet

```
# This code displays the name of the currently running Ignition version number.  
system.gui.messageBox("You are running project: %s" % system.util.getVersion())
```

Code Snippet

```
# This code displays whether a given version is older than the current version.  
currentVersion = system.util.getVersion()  
testVersion = "8.0.7"  
isFuture = currentVersion.isFutureVersion(testVersion)  
print "Your version (%s) is older than %s: %s" %(currentVersion, testVersion, isFuture)
```

Keywords

system util getVersion, util.getVersion

system.util.invokeAsynchronous

This function is used in **Python Scripting**.

Description

Invokes (calls) the given Python function on a different thread. This means that calls to invokeAsynchronous will return immediately, and then the given function will start executing asynchronously on a different thread. This is useful for long-running data intensive functions, where running them synchronously (in the GUI thread) would make the GUI non-responsive for an unacceptable amount of time.

Caution: Under no circumstances should you ever do anything in the function that is invoked asynchronously that interacts with the GUI. This means things like window navigation, setting and getting component properties, showing error/message popups, etc. If you need to do something with the GUI in this function, this must be achieved through a call to [system.util.invokeLaterLater](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.invokeAsynchronous(function, [args], [kwargs], [description])

- Parameters

[Callable](#) function - A Python function object that will be invoked in a newly created thread.

[List \[Any\]](#) args - A list or tuple of Python objects that will be provided to the called function as arguments. Equivalent to the `*` operator. [optional]

[Dictionary\[String, Any\]](#) kwargs - A dictionary of keyword argument names to Python object values that will be provided to the called function as keyword arguments. Equivalent to the `**` operator. [optional]

[String](#) description - A description to use for the asynchronous thread. Will be displayed on the current scope's diagnostic view for scripts. For Vision and the Designer, this would be the "Scripts" tab of the Diagnostics popup. For Perspective and the Gateway scope, this would be the Gateway's [Running Scripts](#) status page. [optional]

- Returns

[Thread](#) - The executing [Thread](#).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would do some data-intensive processing, and then call
# back to the GUI to let it know that it is finished.
# We use default function parameters to pass the root container into these
# functions. (See a Python reference if you don't understand this)

def longProcess(rootContainer = event.source.parent):
    import system
    # Do something here with the database that takes a long time.
    results = ... ( something )
    # Now we'll send our results back to the UI.
    def sendBack(results = results, rootContainer = rootContainer):
        rootContainer.resultsProperty = results
        system.util.invokeLaterLater(sendBack)
```

```
system.util.invokeAsynchronous(longProcess) #Note that this is 'longProcess' instead of 'longProcess()'
```

Keywords

system util invokeAsynchronous, util.invokeAsynchronous

system.util.invokeLater

This function is used in [Python Scripting](#).

Description

Invokes (calls) the given Python function object after all of the currently processing and pending events are done being processed, or after a specified delay. The function will be executed on the GUI, or event dispatch, thread. This is useful for events like propertyChange events, where the script is called before any bindings are evaluated.

If you specify an optional time argument (number of milliseconds), the function will be invoked after all currently processing and pending events are processed plus the duration of that time.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.invokeLater(function, [delay])

- Parameters

[Callable](#) function - A Python function object that will be invoked later, on the GUI, or event-dispatch, thread with no arguments.

[Integer](#) delay - A delay, in milliseconds, to wait before the function is invoked. The default is 0, which means it will be invoked after all currently pending events are processed. [optional]

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# The code in the update/refresh button uses the 'date' property on the two
# calendar components, which are bound to the current_timestamp property on their
# parent. We want to simulate a button press when the window opens, but only
# after the date properties' bindings have been evaluated.

if event.propertyName == 'current_timestamp':
    # Define a function to click the button.
    def clickButton(button = event.source.parent.getComponent('Refresh')):
        import system
        button.doClick()
        system.gui.messageBox("Button has been clicked!")

    # Tell the system to invoke the function after
    # the current event has been processed.
    system.util.invokeLater(clickButton)
```

Keywords

system util invokeLater, util.invokeLater

system.util.jsonDecode

This function is used in [Python Scripting](#).

Description

Takes a JSON string and converts it into a Python object such as a list or a dictionary. If the input is not valid JSON, a string is returned.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.jsonDecode(jsonString)

- Parameters

[String](#) `jsonString` - The JSON string to decode into a Python object.

- Returns

[Any](#) - The decoded Python object. See the table below for a listing of how JSON objects are mapped to Python objects.

- Scope

Gateway, Vision Client, Perspective Session

JSON to Python Mapping

The table below lists possible JSON types, and the Python types this function maps to.

JSON Type	Mapped Python Type
Boolean (true/false)	Boolean (True/False)
String	String
Numeric	Number (Float, Integer)
null	None
Array	List
Object	Dictionary

Code Examples

```
# The following example reads in a JSON string, and converts the string to a Python object.  
# The example attempts to read the JSON string from a text file, but this could easily be modified to  
read data from a web server.  
  
# Read the JSON string.  
jsonString = system.file.readFileAsString("C:\\\\tmp\\\\json.txt")  
  
# Decode the JSON string and store the results into a variable.  
obj = system.util.jsonDecode(jsonString)  
  
# Do something with the results. The code below prints the data type of the results to the console.  
print type(obj)
```

Keywords

system util jsonDecode, util.jsonDecode

system.util.jsonEncode

This function is used in [Python Scripting](#).

Description

Takes a Python object such as a list or dictionary and converts into a JSON string.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.jsonEncode(pyObj, [indentFactor])

- Parameters

[Any](#) pyObj - The Python object to encode into JSON such as a Python list or dictionary.

[Integer](#) indentFactor - The number of spaces to add to each level of indentation for prettyprinting. [optional]

- Returns

[String](#) - The encoded JSON string.

- Scope

Gateway, Vision Client, Perspective Session

JSON to Python Mapping

The table below lists possible Python types, and how they map to JSON objects.

Python Type	Mapped JSON Type
Boolean (True/False)	Boolean (true/false)
String	String
Number (Float, Integer)	Numeric
None	null
Sequence	Array
Dictionary	Object

Code Examples

Code Snippet

```
# The following example builds a Python dictionary and converts it to a JSON string.

# Build the Python dictionary.
employeeDict = {"employees": [{"firstName": "John", "lastName": "Doe"}, {"firstName": "Anna", "lastName": "Smith"}, {"firstName": "Peter", "lastName": "Jones"}]}

# Convert the dictionary and store the resulting JSON string in a variable.
jsonString = system.util.jsonEncode(employeeDict)
```

Keywords

system util jsonEncode, util.jsonEncode

system.util.modifyTranslation

This function is used in **Python Scripting**.

Description

This function allows you to add or modify a global translation.

Client Permission Restrictions

Permission Type: Translation Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.util.modifyTranslation(term, translation, [locale])

- Parameters

String term - The key term to translate.

String translation - The translated value to store.

String locale - If specified, the locale code (such as "es") identifying the language of the translation. If omitted, the function will attempt to detect the locale automatically. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code adds or updates a translation into French
# for the word Hello. Note the u in front "Allô!", which
# is needed for Python strings outside of the 7-bit ASCII
# range.
system.util.modifyTranslation("Hello", u"Allô!", "fr")
```

Keywords

system util modifyTranslation, util.modifyTranslation

system.util.playSoundClip

This function is used in [Python Scripting](#).

Description

Plays a sound clip from a wav file to the system's default audio device. The wav file can be specified as a filepath, a URL, or directly as a raw List [Byte].

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.playSoundClip(wavBytes, [volume], [wait])

- Parameters

[List\[Byte\]](#) wavBytes - A byte list of a wav file.

[Float](#) volume - The clip's volume, represented as a floating point number between 0.0 and 1.0. [optional]

[Boolean](#) wait - A boolean flag indicating whether or not the call to playSoundClip should wait for the clip to finish before it returns. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Syntax

system.util.playSoundClip(wavFile, [volume], [wait])

- Parameters

[String](#) wavFile - A filepath or URL that represents a wav file.

[Float](#) volume - The clip's volume, represented as a floating point number between 0.0 and 1.0. [optional]

[Boolean](#) wait - A boolean flag indicating whether or not the call to playSoundClip should wait for the clip to finish before it returns. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code plays a sound clip at full volume that was located on the current
# host's filesystem. It will not return until the clip is finished playing.
system.util.playSoundClip("C:\\sounds\\siren.wav")
```

Code Snippet

```
# This code would pull a sound clip out of a BLOB field from a database,  
# playing it asynchronously at half volume.  
  
query = "SELECT wavBlob FROM sounds WHERE type='alert_high'"  
soundData = system.db.runScalarQuery(query)  
  
system.util.playSoundClip(soundData, 0.5, 0)
```

Keywords

system util playSoundClip, util.playSoundClip

system.util.queryAuditLog

This function is used in [Python Scripting](#).

Description

Queries an audit profile for audit history. Returns the results as a dataset.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.util.queryAuditLog([auditProfileName], [startDate], [endDate], [actorFilter], [actionFilter], [targetFilter], [valueFilter], [systemFilter], [contextFilter])
```

- Parameters

[String](#) auditProfileName - The name of the audit profile to pull the history from. [optional]
[Date](#) startDate - The earliest audit event to return. If omitted, the current time - 8 hours will be used. [optional]
[Date](#) endDate - The latest audit event to return. If omitted, the current time will be used. [optional]
[String](#) actorFilter - A filter string used to restrict the results by actor. [optional]
[String](#) actionFilter - A filter string used to restrict the results by action. [optional]
[String](#) targetFilter - A filter string used to restrict the results by target. [optional]
[String](#) valueFilter - A filter string used to restrict the results by value. [optional]
[String](#) systemFilter - A filter string used to restrict the results by system. [optional]
[Integer](#) contextFilter - A bitmask used to restrict the results by context. 0x01 = Gateway, 0x02 = Designer, 0x04 = Client. [optional]

- Returns

[Dataset](#) - A dataset with the audit events from the specified profile that match the filter arguments.

- Scope

Gateway, Vision Client

Code Examples

```
# This script queries an audit log, checks to see if a user john made any tag writes in the last 8 hours,
# and writes the results to a table.

data = system.util.queryAuditLog(auditProfileName='AuditLog', actorFilter='john', actionFilter='tag
write')

event.source.parent.getComponent("Table").data = data
```

Keywords

system util queryAuditLog, util.queryAuditLog

system.util.retarget

This function is used in **Python Scripting**.

Description

This function allows you to programmatically 'retarget' the Client to a different project and/or different Gateway. You can have it switch to another project on the same Gateway, or another Gateway entirely, even across a WAN. This feature makes the vision of a seamless, enterprise-wide SCADA application a reality.

The retarget feature will attempt to transfer the current user credentials over to the new project / Gateway. If the credentials fail on that project, the user will be prompted for a valid username and password. Once valid authentication has been achieved, the currently running project is shut down, and the new project is loaded.

You can pass any information to the other project through the parameters dictionary. All entries in this dictionary will be set in the global scripting namespace in the other project. Even if you don't specify any parameters, the system will set the variable `_RETARGET_FROM_PROJECT` to the name of the current project and `_RETARGET_FROM_GATEWAY` to the address of the current Gateway.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

Note: This function accepts [keyword arguments](#).

system.util.retarget(project, [addresses], [params], [windows])

- Parameters

String project - The name of the project to retarget to.

String or List addresses - The address of the Gateway that the project resides on. If omitted, the current Gateway will be used. Format is: **host:port**. This can be a list of strings. When using a list, the function will try each address in order, waiting for the timeout period between each address attempt. [optional]

Dictionary[String, Any] params - A dictionary of parameters that will be passed to the new project. They will be set as global variables in the new project's Python scripting environment. [optional]

List[String] windows - A list of window paths to use as the startup windows. If omitted, the project's normal startup windows will be opened. If specified, the project's normal startup windows will be ignored, and this list will be used instead. [optional]

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code would switch to a project named "TankControl" on the same Gateway  
# as the currently running project.  
system.util.retarget("TankControl")
```

Code Snippet

```
# This code would switch to a project named "TankControl" on a
# Gateway located at a different IP address running on port 8080, and
# would open the window named "Graph", and set a global JYTHON variable in the
# new project named "retargetOccured" to the value 1 (one).
system.util.rettarget("TankControl", "10.30.2.33:8088", {"retargetOccured":1}, ["Graph"])
```

Code Snippet

```
# This code would switch to a project named "TankControl" on a
# Gateway located at a different IP address using SSL on port 8043.
system.util.rettarget("TankControl", "10.30.2.34:8043")
```

Code Snippet

```
# This code would be put in a button in the target that was retargeted to,
# and act as a 'back' button, that would retarget back to the original project.

# Fetch the global values that are automatically created when you retarget.
project = system.util.getGlobals()['_RETARGET_FROM_PROJECT']
gateway = system.util.getGlobals()['_RETARGET_FROM_GATEWAY']

# Retarget.
system.util.rettarget(project, gateway)
```

Keywords

system util rettarget, util.rettarget

system.util.sendMessage

This function is used in [Python Scripting](#).

Description

This function sends a message to clients running under the Gateway or to a project within the Gateway itself. To handle received messages, you must set up event script message handlers within a project. These message handlers run Jython code when a message is received. You can add message handlers under the "Message" section of the client/Gateway event script configuration dialogs.

Messages cannot be received within a Designer. However, messages can be sent within the Designer in a script (assuming that read/write comm is enabled).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.util.sendMessage(project, messageHandler, [payload], [scope], [clientSessionId], [user], [hasRole], [hostName], [remoteServers])`

- Parameters

`String` `project` - The name of the project containing the message handler.

`String` `messageHandler` - The name of the message handler that will fire upon receiving a message.

`Dictionary[String, Any]` `payload` - A dictionary which will get passed to the message handler. Use "payload" in the message handler to access dictionary variables. [optional]

`String` `scope` - Limits the scope of the message delivery to "C" (clients), "G" (Gateway), "CG" for clients and the Gateway, or "S" Session. Any combination of C, G, and S are available. Defaults to "CS" if the user name, role, or host name parameters are set, and to "CGS" if none of these parameters are set. [optional]

`String` `clientSessionId` - Limits the message delivery to a client with the specified session ID. [optional]

`String` `user` - Limits the message delivery to clients where the specified user has logged in. [optional]

`String` `hasRole` - Limits the message delivery to any client where the logged in user has the specified user role. [optional]

`String` `hostName` - Limits the message delivery to the client that has the specified network host name. [optional]

`List` `remoteServers` (since 7.8.2) - A list of Strings representing Gateway Server names. The message will be delivered to each server in the list. Upon delivery, the message is distributed to the local Gateway and clients as per the other parameters. [optional]

- Returns

`List` - A List of strings containing information about each system that was selected for delivery, where each List item is comma-delimited.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Simple message to both Client and Gateway handlers.  
project="X"  
# It's important that both Gateway and Client versions of this message handler have been created.  
messageHandler="myMessageHandler"  
scope="CG"  
myDict = {'first': "Hello", 'second': "World"}  
results=system.util.sendMessage(project,messageHandler,myDict,scope)
```

```
# Assuming that there is one local client running project X, the results List will contain these Strings:  
#type=Gateway,project=X,messageHandler=testHandler,filterParams={hostName=, clientSessionId=, scope=CG,  
user=, hasRole=},sendStatus=SENT  
  
#type=Client,sessionId=65F7A472,clientAddress=127.0.0.1,clientHostName=127.0.0.1,project=X,  
messageHandler=testHandler,filterParams={hostName=, clientSessionId=, scope=CG, user=, hasRole=},  
sendStatus=SENT
```

Code Snippet

```
# Message to client handlers only where a specified user is logged in)  
system.util.sendMessage(project="X",messageHandler="myMessageHandler",scope="C",user="Bob")
```

Code Snippet

```
# Message to remote servers over the Gateway Network (since 7.8.2)  
servers = ["agent-8088", "agent-9000"]  
system.util.sendMessage(project="X",messageHandler="myMessageHandler",remoteServers=servers)
```

Keywords

system util sendMessage, util.sendMessage

system.util.sendRequest

This function is used in **Python Scripting**.

Description

This function sends a message to the Gateway, working in a similar manner to the [sendMessage](#) function, except sendRequest expects a response to the message. To handle received messages, you must set up Gateway Event Script message handlers within a project. These message handlers run Jython code when a message is received. You can then place a return at the end of the code to return something to where the sendRequest was originally called from. You can add message handlers under the "Message" section of the Gateway Event Script configuration dialog.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.sendRequest(project, messageHandler, [payload], [remoteServer], [timeoutSec])

- Parameters

[String](#) project - The name of the project containing the message handler.

[String](#) messageHandler - The name of the message handler that will fire upon receiving a message.

[Dictionary\[String, Any\]](#) payload - A dictionary which will get passed to the message handler. Use "payload" in the message handler to access dictionary variables. [optional]

[String](#) hostName - Limits the message delivery to the client that has the specified network host name. [optional]

[String](#) remoteServer - A string representing a target Gateway Server name. The message will be delivered to the remote Gateway over the Gateway Network. Upon delivery, the message is distributed to the local Gateway and clients as per the other parameters. [optional]

[String](#) timeoutSec - The number of seconds before the sendRequest call times out. [optional]

- Returns

[Object](#) - The return from the message handler.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Request sent to the message handler 'test' which then saves the return value to returnValue and prints it.  
returnValue = system.util.sendRequest(project='ACME', messageHandler='test', payload={'hoursOn':15})  
print returnValue
```

Keywords

system util sendRequest, util.sendRequest

system.util.sendRequestAsync

This function is used in [Python Scripting](#).

Description

This function sends a message to the Gateway and expects a response. Works in a similar manner to the [sendRequest](#) function, except sendRequestAsync will send the request and then immediately return a handle for it. The Request handle has the following methods:

- `get()` - Block for result, throw an exception on failure.
- `cancel()` - Cancel the request. Any completion callback will be called with `CancellationException`
- `block()` - Like `get()`, but will return a Boolean True or False once complete, indicating completion success. If False, call `getError()` to get the exception object.
- `getError()` - Returns the error result or null. Similar to `get()`, in that this will block for a result.
- `onSuccess(PyFunction)` - Will set a function to run on a successful completion callback or set a new one if one was already defined in the `sendRequestAsync` call.
- `onError(PyFunction)` - Will set a function to run on a failed completion callback or set a new one if one was already defined in the `sendRequestAsync` call.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.util.sendRequestAsync(project, messageHandler, [payload], [remoteServer], [timeoutSec], [onSuccess], [onError])`

- Parameters

`String project` - The name of the project containing the message handler.

`String messageHandler` - The name of the message handler that will fire upon receiving a message.

`Dictionary[String, Any] payload` - A dictionary which will get passed to the message handler. Use "payload" in the message handler to access dictionary variables. [optional]

`String hostName` - Limits the message delivery to the client that has the specified network host name. [optional]

`String remoteServer` - A string representing the target Gateway server name. The message will be delivered to the remote Gateway over the Gateway Network. Upon delivery, the message is distributed to the local Gateway and clients as per the other parameters. [optional]

`String timeoutSec` - The number of seconds before the `sendRequest` call times out. [optional]

`Callable onSuccess` - Should take one argument, which will be the result from the message handler. Callback functions will be executed on the GUI thread, similar to [system.util.invokeLater](#). [optional]

`Callable onError` - Should take one argument, which will be the exception encountered. Callback functions will be executed on the GUI thread, similar to [system.util.invokeLater](#). [optional]

- Returns

`Request Handle` - The [Request](#) object that can be used while waiting for the message handler callback.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This calls the message handler 'test', and returns a handle into myHandle.  
# We then call get() on myHandle, which will block the script and will wait for a return or throw an  
exception on failure.
```

```
myHandle = system.util.sendRequestAsync(project='ACME', messageHandler='test', payload={'number':55})  
myHandle.get()
```

```
# This will call the message handler 'test', and will return a handle into myHandle.  
# In this example, we will define a function to run when the message handler has successfully finished,  
# using the onSuccess function on the Request Handle.  
  
# Note that function accepts a single argument for the message.  
def successFunc(message):  
    system.gui.messageBox("Successfully finished: %s" % message)  
  
# We're specifying that the request should timeout after 5 seconds.  
myHandle = system.util.sendRequestAsync(project='ACME', messageHandler='test', payload={'number':55},  
timeoutSec=5)  
  
# Call the Request Handler's onSuccess function, passing in successFunc.  
myHandle.onSuccess(successFunc)
```

Keywords

system util sendRequestAsync, util.sendRequestAsync

system.util.setConnectionMode

This function is used in [Python Scripting](#).

Description

Sets the connection mode for the Client. Normally a Client runs in mode 3, which is read-write. You may wish to change this to mode 2, which is read-only, which will only allow reading and subscribing to Tags, and running SELECT queries. Tag writes and INSERT / UPDATE / DELETE queries will not function. You can also set the connection mode to mode 1, which is disconnected, all Tag and query features will not work.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.setConnectionMode(mode)

- Parameters
 - Integer mode** - The new connection mode: 1 = Disconnected, 2 = Read-only, 3 = Read/Write.
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

Code Snippet

```
# This example, which could go in a project's startup script, checks the current username
# and sets the connection mode to read-only if it is the "guest" user.

username = system.security.getUsername()
if "guest" == username.lower():
    # Set "guest" user to read-only mode.
    system.util.setConnectionMode(2)
else:
    system.util.setConnectionMode(3)
```

Keywords

system util setConnectionMode, util.setConnectionMode

system.util.setConnectTimeout

This function is used in [Python Scripting](#).

Description

Sets the connect timeout for Client-to-Gateway communication. Specified in milliseconds.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.setConnectTimeout(connectTimeout)

- Parameters

[Integer](#) connectTimeout - The new connect timeout, specified in milliseconds.

- Returns

Nothing

- Scope

Vision Client

Code Examples

Code Snippet

```
# This code sets the current connect timeout to 30 seconds.  
system.util.setConnectTimeout(30000)
```

Keywords

system util setConnectTimeout, util.setConnectTimeout

system.util.setLocale

This function is used in [Python Scripting](#).

Description

Sets the user's current locale. Any valid Java locale code (case-insensitive) can be used as a parameter, including ones that have not yet been added to the Translation Manager. An invalid locale code will cause an Illegal Argument Exception.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.setLocale(locale)

- Parameters
 - String locale - A locale code, such as 'en_US' for US English.
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

```
# This script will set the client locale to Arabic.  
system.util.setLocale('ar')
```

Keywords

system util setLocale, util.setLocale

system.util.setLoggingLevel

This function is used in [Python Scripting](#).

Description

Sets the logging level on the given logger. This can be a logger you create, or a logger already defined in the system.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.setLoggingLevel(loggerName, loggerLevel)

- Parameters

String loggerName - The unique name of the logger to change the logging level on, for example "Tags.Client".

String loggerLevel - The level you want to change to logger to: "trace", "debug", "info", "warn", or "error".

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This sets the logger called Reporting to the debug level.  
system.util.setLoggingLevel("Reporting", "debug")
```

Keywords

system util setLoggingLevel, util.setLoggingLevel

system.util.setReadTimeout

This function is used in [Python Scripting](#).

Description

Sets the read timeout for Client-to-Gateway communication. Specified in milliseconds.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.setReadTimeout(readTimeout)

- Parameters
 - [Integer](#) readTimeout - The new read timeout, specified in milliseconds.
- Returns
 - Nothing
- Scope
 - Vision Client

Code Examples

```
# This script sets the read timeout to 20 seconds.  
system.util.setReadTimeout(20000)
```

Keywords

system util setReadTimeout, util.setReadTimeout

system.util.threadDump

This function is used in [Python Scripting](#).

Description

Creates a thread dump of the current running JVM.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.util.threadDump()

- Parameters
 - Nothing
- Returns
 - String** The dump of the current running JVM.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

```
# This script takes a thread dump of the current JVM and writes it to a Text Area component.  
event.source.parent.getComponent('Text Area').text = system.util.threadDump()
```

Keywords

system util threadDump, util.threadDump

system.util.translate

This function is used in **Python Scripting**.

Description

This function allows you to retrieve the global translation of a term from the translation database using the current locale.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

Note: This function accepts [keyword arguments](#).

system.util.translate(term, [locale], [strict])

- Parameters

[String](#) term - The term to look up.

[String](#) locale - Which locale to translate against. Useful when there are multiple locales defined for a single term. If omitted, the function attempts to use the current locale (as defined by the client, session, or Designer). [optional]

[Boolean](#) strict - If false, the function will return the passed term (param 1) if it could not find a defined translation for the locale: meaning, if you pass a term that hasn't been configured, the function will just send the term back to you. If true, then the function will return a None when it fails to find a defined translation. Default is false. [optional]

- Returns

[String](#) - The translated term.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

```
# This script will take a term written into a Text Field component, translate it using the translation database, and then write it back to the same Text Field.  
# it uses the current locale since none is specified.
```

```
text = event.source.parent.getComponent('Text Field').text  
translation = system.util.translate(text)  
event.source.parent.getComponent('Text Field').text = translation
```

Python - Picking a Local

```
# This code block demonstrates how to use the locale parameter.  
  
# Use the currently detected locale.  
system.util.translate("Hello")  
  
# Translate to Italian.  
system.util.translate("Hello", "it")
```

```
# Translate to a regional variant - Irish English in this case.  
system.util.translate("Hello", "en-ie")
```

Keywords

system util translate, util.translate

Reference Pages

This section is for various reference pages that may fall outside of a specific section in the User Manual.

[In This Section ...](#)

Color Selector Reference

Color Options in Ignition

The Perspective module, Vision module, and Report module each have color options for customizing the way your projects look. The Color Selector popup has four tabs with options for setting color:

- Color Wheel
- Color Palette
- RGB Color
- HSL Color

Each of these four selectors will give you different options for selecting a single color for your properties. You can combine these in interesting ways with your styles to change foreground, background, border, and many properties. If you want to find multiple colors to create matching or complimentary color selections, you can easily find online color selectors like the one from Sessions College to get color codes and complimentary colors based on a simple selector.

On this page ...

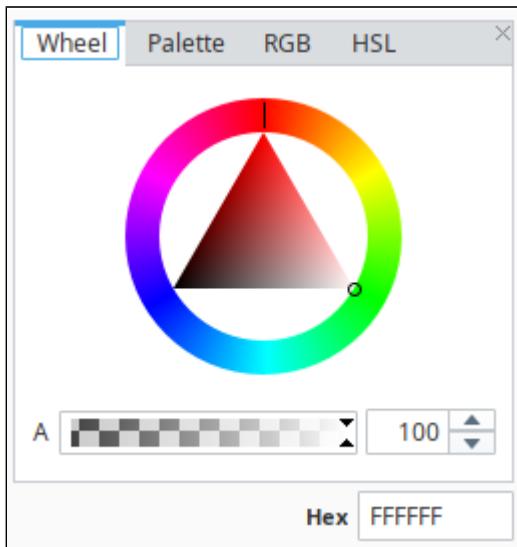
- [Color Options in Ignition](#)
- [Color Wheel](#)
- [Color Palette](#)
 - [Exporting the Palette](#)
 - [Importing a Palette](#)
- [RGB Color](#)
- [HSL Color](#)

Color Wheel

The Wheel tab brings up the Color Wheel. There are a few distinct areas to make selections to set up your color.

- The outer ring shows the color spectrum similar to the Hue in a HSL color selection. You can click and drag anywhere in the outer right to select your hue.
- The inner triangle lets you choose shade of the color selected in the outer ring. The edges of the triangle represent the luminosity, and moving toward the center represents saturation.
- The Alpha slider and value allow you to set the transparency level. 0 is completely transparent, and 100 is completely opaque.
- The Hex value representing the RGB color will be shown at the bottom of the tab. You can manually change this or see it update as you change your selections above. If you change the Alpha, a fourth set of values will be added to the Hex.

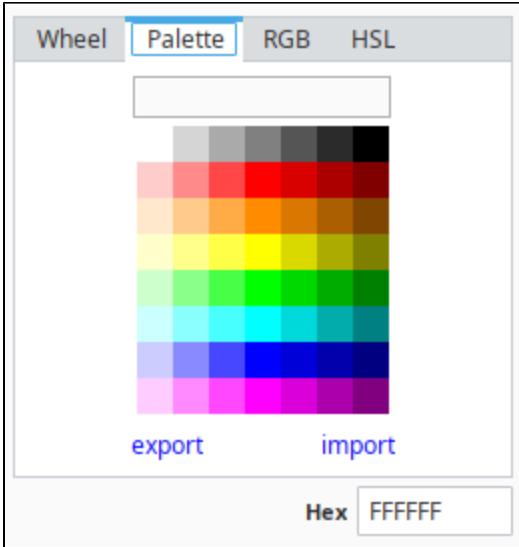
To find a color, first click and drag the line in the outer ring. Next click within the triangle.



Color Palette

The Color Palette tab offers a more simple view of colors. Just click on the color you want for your item. The Hex equivalent is also given at the bottom of the palette.

Recently selected colors will show in the row across the top of the palette, and you can import/export your palettes if you have a custom color scheme.



Exporting the Palette

You can export the Color Palette by clicking the export button in the bottom left. The file will be saved as a .CSV file type. You can edit the file in any text editor and import it back into Ignition.

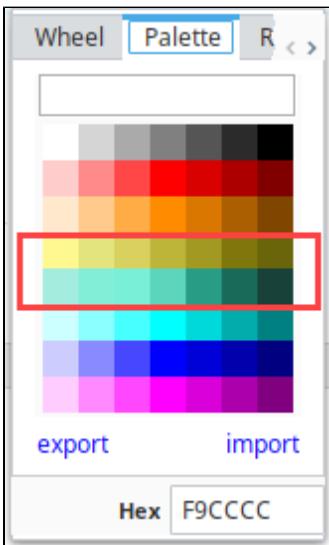
The .CSV file for the standard Color Palette in Ignition is as follows. Each color has a Hex RGB (red, green, blue) or RGBA (red, green, blue, alpha) value in quotation marks and is separated by commas. You can save this file as a .txt file or a .CSV and Ignition will be able to import it. Add as many rows or columns as you want.

```
"#FFFFFF", "#D5D5D5", "#AAAAAA", "#808080", "#555555", "#2B2B2B", "#000000"
"#FFCCCC", "#FF8A8A", "#FF4747", "#FF0000", "#D90000", "#AC0000", "#800000"
"#FFE8CC", "#FFCA8A", "#FFAC47", "#FF8C00", "#D97700", "#AC5F00", "#804600"
"#FFFFCC", "#FFF8A", "#FFF47", "#FFF00", "#D9D900", "#ACAC00", "#808000"
"#CCFFCC", "#8AFF8A", "#47FF47", "#00FF00", "#00D900", "#00AC00", "#008000"
"#CCFFFF", "#8AFFFF", "#47FFFF", "#00FFFF", "#00D9D9", "#00ACAC", "#008080"
"#CCCCFF", "#8A8AFF", "#4747FF", "#0000FF", "#0000D9", "#0000AC", "#000080"
"#FFCCFF", "#FF8AFF", "#FF47FF", "#FF00FF", "#D900D9", "#AC00AC", "#800080"
```

Importing a Palette

You can also import existing .CSV files to the Color Palette. The Palette import function will also accept a .txt file if it is formatted correctly.

In this example, we modified the colors in the fourth and fifth rows of the original palette.

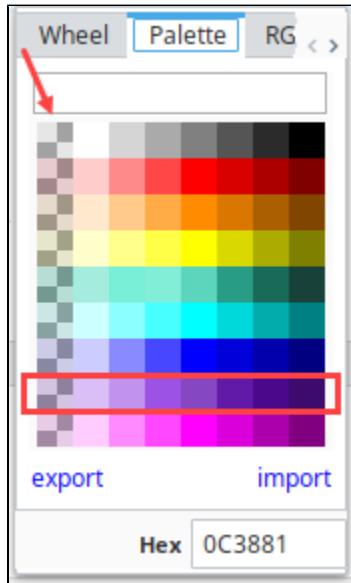


You can also import a new palette as long as the file contains Hex color values in quotation marks, separated by commas. The Color Palette is not limited in rows or columns. You can import a .CSV file with more colors than the standard palette in Ignition.

In this example, we added column with a 50% opacity of the first color in each row. We also added a new row with colors in the purple color family.

Each color has a Hex RGB (red, green, blue) or RGBA (red, green, blue, alpha) value in quotation marks and is separated by commas. Add as many rows or columns as you want.

```
"FFFFFF80", "#FFFFFF", "#D5D5D5", "#AAAAAA", "#808080", "#555555", "#2B2B2B", "#000000"
"FFCCCC80", "#FFCCCC", "#FF8A8A", "#FF4747", "#FF0000", "#D90000", "#AC0000", "#800000"
"FFE8CC80", "#FFE8CC", "#FFCA8A", "#FFAC47", "#FF8C00", "#D97700", "#AC5F00", "#804600"
"FFFFCC80", "#FFFFCC", "#FFFF8A", "#FFF47", "#FFF00", "#D9D900", "#ACAC00", "#808000"
"A4EDDE80", "#A4EDDE", "#79EFD7", "#82EED8", "#5BD5BC", "#299C85", "#1A6A5A", "#174139"
"CCFFFF80", "#CCFFFF", "#8AFFFF", "#47FFFF", "#00FFFF", "#00D9D9", "#00ACAC", "#008080"
"CCCCFF80", "#CCCCFF", "#8A8AFF", "#4747FF", "#0000FF", "#0000D9", "#0000AC", "#000080"
"DABEF680", "#DABEF6", "#C193EE", "#9C52E5", "#8647C3", "#601AA7", "#4A088B", "#3C0B6D"
"FFCCFF80", "#FFCCFF", "#FF8AFF", "#FF47FF", "#FF00FF", "#D900D9", "#AC00AC", "#800080"
```



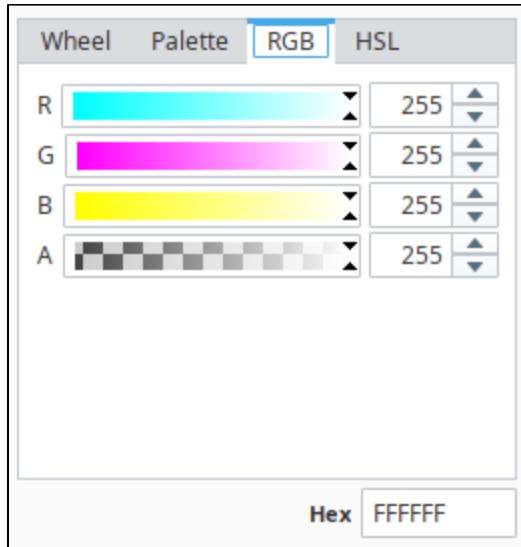
RGB Color

The third tab on the Color options is RGBA (red, green, blue, and alpha). RGB refers to a system for representing the colors to be used on a computer display. Red, green, and blue can be combined in various proportions to obtain any color in the visible spectrum. The color is specified by an RGB value, commonly expressed in either of two ways. The decimal format is a comma separated string with one number for each color ranging from 0 to 255. For example, "0,255,0" represents green. The other is a Hex format starting with a # symbol and then listing two-digit hex values for each color, ranging from 00 to FF. For example, "#00FF00" represents green.

RGBA color values are an extension of RGB color values with an alpha channel, which specifies the opacity for a color. An RGBA color value is specified with: **rgba(red, green, blue, alpha)**. In scripting the alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque). In this color selector, it follows the same rules as the other colors.

In the RGB tab, you can change the color by either dragging the selector inside the color box, using the up and down arrows next to a color box, or typing in a value.

The Hex value for the color will be shown at the bottom of the tab.

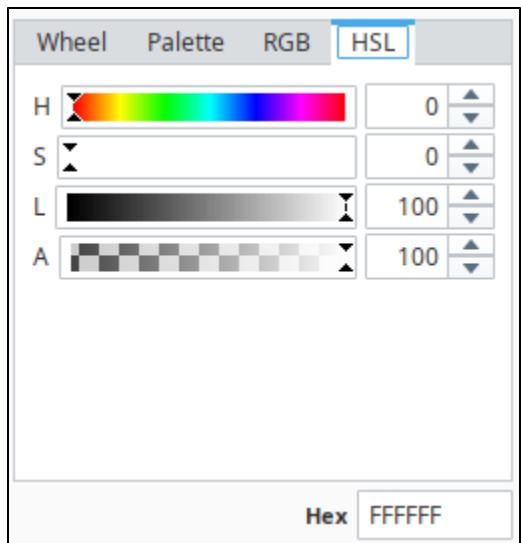


HSL Color

The third tab on the Color options is HSL (hue, saturation, and luminosity). HSL settings are used to adjust the way a color is displayed. Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, 240 is blue. Saturation is a percentage value; 0% means a shade of gray and 100% is the full color. Luminosity is also a percentage; 0% is black, 50% is the selected Hue, and 100% is white. Alpha is a percentage value; 0% means completely transparent and 100% is a solid color based on the other three options.

In the HSL tab, you can change the color by either dragging the selector inside the color box, using the up and down arrows next to a color box, or typing in a value.

The Hex value for the color will be shown at the bottom of the tab.



Gateway Configuration File Reference

This section details the various configuration changes that can be made to the Gateway's configuration file: `ignition.conf`. Before the Gateway starts, a wrapper service checks the configuration file for parameter values, and then attempts to start the Gateway using parameters. This means changes made to the configuration file will only occur on startup, and making changes to the configuration file while the Gateway is already running will not have any impact until a restart.

Configuration File Location

The Gateway's configuration file can be found at:

`%IgnitionInstallationDirectory%/data/ignition.conf`.

See the [Installing and Upgrading Ignition](#) page for default installation directories.

Comments in the File

Many lines in the configuration file are commented out, either to provide some documentation in the file, or to omit certain parameters, such as some options under the Java Additional Parameters header. The "`#`" character is used to comment lines in the configuration file.

```
# this line is commented  
this line is not commented
```

On this page ...

- [Configuration File Location](#)
- [Comments in the File](#)
- [Gateway Memory Allocation](#)
- [Changing Java Additional Parameters](#)
- [Gateway Security HTTP Headers and Configuration](#)
 - [Strict Transport Security](#)
 - [Referrer Policy](#)
 - [X Content Type Options](#)
 - [X Frame Options](#)
 - [X XSS Protection](#)
- [Perspective Web Socket Compression](#)
- [Maximum Form Size](#)
- [Hosted Launcher Installers](#)
 - [Enabling Hosted Launchers](#)
 - [Hosted Launcher Version](#)
- [Ignition Edition](#)
 - [Values](#)
- [Tag Historian](#)
 - [Tag Historian Query Syntax](#)
- [Store and Forward](#)

Gateway Memory Allocation

On a standard installation, the Gateway is initially allocated a fixed amount of heap (memory). Increasing this allows the Gateway to utilize more of the server's memory. To determine if this is necessary, navigate to the [Performance](#) page under the Status section of the Gateway webpage. Generally speaking, if the Memory Trend shows a saw tooth pattern that peaks towards the maximum amount of memory, that's expected behavior. However, if memory usage usually floats around maximum allocated, and slow response events occur more than occasionally, then it may be time to consider increasing the maximum allocated memory, assuming the server has available memory.

To increase the maximum amount of memory allocated to the Gateway, open the Gateway configuration file, and search for "`wrapper.java.maxmemory`". It should look something like following:

```
# Maximum Java Heap Size (in MB)  
wrapper.java.maxmemory=1024
```

The number at the end of the line represents the maximum amount of memory allocated to the Gateway (in megabytes). To change the amount of memory allocated to the Gateway, simply change this number, save the configuration file, and restart the Gateway. Once the Gateway starts up again, it will attempt to use the amount of memory specified.

The new heap size shouldn't exceed the amount of memory available, nor should you allocate all of the server's memory to the Gateway because the server likely has other applications (including the host operating system) that also need to make use of the system's resources.



Changing Gateway Memory Allocation

[Watch the Video](#)

Changing Java Additional Parameters

Caution:

Adding or modifying parameters in the configuration file is considered an advanced configuration change. Most installations don't require any additional parameters, nor do they require modification to existing parameters. We generally discourage most users from making changes to parameters in the configuration file, as doing so could result in some unintended behavior or security vulnerabilities. We list the parameters on this page for the sake of transparency.

If you choose to add or modify parameters in the configuration file, you do so at your own risk.

A section of the configuration file contains a header stating "Java Additional Parameters". This section allows for a large number of configuration changes, and merits having some discussion on how to add new parameters. On install, the Java Additional Parameters section may look like the following:

```
# Java Additional Parameters
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#Wrapper.java.additional.3=-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=*:8000
```

When adding a new parameter, the "wrapper.java.additional.#" prefix must be added to a new line. Each parameter contains a **prefix** ("wrapper.java.additional.#"), a **key**, and a **value** that the key will be set to. Generally speaking, each parameter in the file should follow the pattern below:

```
wrapper.java.additional.1=-key=value
```

Uncommented parameters should be listed in ascending numerical order, based on the number at the end of the prefix, as shown below.

```
# Add parameters like this:
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#Wrapper.java.additional.3=-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=*:8000
wrapper.java.additional.2=-some.thing=foo
wrapper.java.additional.3=-some.other.thing=bar

# Avoid skipping commented numbers:
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#Wrapper.java.additional.3=-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=*:8000
wrapper.java.additional.4=-some.thing=foo
```

Gateway Security HTTP Headers and Configuration

Gateways use more secure security headers. The default settings are listed below, along with their associated keys. These parameters mainly impact Perspective sessions, as well as any pages hosted on Ignition's web server.

Note: HTTP headers used by the Gateway are configurable via the ignition.conf file in the Gateway's installation directory. In most cases there won't be a need to modify the new default values. However, if you're embedding a Perspective session inside of an iframe from another origin, and it stops working after upgrade, then take a look at this [Knowledge Base Article](#) for more details.

Strict Transport Security

The following keys interact with the [Strict-Transport-Security](#) header. Enabling the header involves setting the Dignition.https.sts.maxAge key to a non-zero value. By default, the header is disabled.

Key	Description
-Dignition.https.sts.maxAge	Sets the maximum age of the Strict Transport Security header. The value should be set to an integer, representing a number of seconds.
-Dignition.https.sts.includeSubDomains	Applies the includeSubDomains parameter. Expects a boolean value. If omitted, defaults to false.

Example

```
wrapper.java.additional.1=-Dignition.https.sts.maxAge=5000  
wrapper.java.additional.2=-Dignition.https.sts.includeSubDomains=false
```

Referrer Policy

The following keys interact with the [Referrer-Policy](#) header: By default, the header is enabled with a value of `strict-origin-when-cross-origin`.

Key	Description
<code>-Dignition.http.header.referrer_policy.enabled</code>	Allows you to enable or disable the Referrer-Policy header. Expects a <code>true</code> or <code>false</code> value.
<code>-Dignition.http.header.referrer_policy.value</code>	Sets the value of the Referrer-Policy.

Example

```
wrapper.java.additional.1=-Dignition.http.header.referrer_policy.enabled=true  
wrapper.java.additional.2=-Dignition.http.header.referrer_policy.value=strict-origin-when-cross-origin
```

X Content Type Options

The following keys interact with the [X-Content-Type-Options](#) header: By default, the header is enabled with a value of `nosniff`.

Key	Description
<code>-Dignition.http.header.x_content_type_options.enabled</code>	Allows you to enable or disable the X-Content-Type-Options header. Expects a <code>true</code> or <code>false</code> value.
<code>-Dignition.http.header.x_content_type_options.value</code>	Determines the value of the X-Content-Type-Options.

Example

```
wrapper.java.additional.1=-Dignition.http.header.x_content_type_options.enabled=true  
wrapper.java.additional.2=-Dignition.http.header.x_content_type_options.value=nosniff
```

X Frame Options

The following keys interact with the [X-Frame-Options](#) header: By default, the header is enabled with a value of `SAMEORIGIN`.

Key	Description
<code>-Dignition.http.header.x_frame_options.enabled</code>	Enables or disables the X-Frame-Options header.
<code>-Dignition.http.header.x_frame_options.value</code>	Determines the value of the X-Frame-Options header

Example

```
wrapper.java.additional.1=-Dignition.http.header.x_frame_options.enabled=true  
wrapper.java.additional.2=-Dignition.http.header.x_frame_options.value=SAMEORIGIN
```

X XSS Protection

The following keys interact with the [X-XSS-Protection](#) header. By default, the header is enabled with a value of 1; mode=block.

Key	Description
-Dignition.http.header.x_xss_protection.enabled	Enables or disables the X-XSS-Protection header.
-Dignition.http.header.x_xss_protection.value	Determines the value of the X-XSS-Protection header.

Example

```
wrapper.java.additional.1=-Dignition.http.header.x_xss_protection.enabled=true  
wrapper.java.additional.2=-Dignition.http.header.x_xss_protection.value=1; mode=block
```

Perspective Web Socket Compression

You can remove the compression header for perspective web socket connections.

This parameter is generally only used in cases where a Microsoft IIS based firewall is acting like a reverse proxy, and needs to be able to rewrite secure web-socket messages.

Example

```
wrapper.java.additional.##=-Dperspective.ws.disablePermessageDeflateExtension=true
```

Maximum Form Size

Note: By default, the maximum size is kept low on the default installation to prevent DoS attacks. We do not recommend changing this, especially so if the Gateway isn't in a closed network.

This parameter allows you limit the amount of data that can be posted back from a browser or other client, to the Gateway. Generally, this is useful to increase the maximum size for [WebDev](#) POST requests. More information about this parameter can be found in [Jetty's documentation](#).

Example

```
wrapper.java.additional.##=-Dorg.eclipse.jetty.server.Request.maxFormContentSize=2000000
```

Hosted Launcher Installers

Normally, each Ignition Gateway includes files for the various launchers. When you download a launcher from a Gateway, it simply streams its local launcher files. However, you can override this behavior, causing the Gateway to ignore its local launcher files and instead download launchers from

the internet. This is an uncommon configuration for most cases, as it was devised primarily to aid with systems where memory limitations are strict (such as physical devices that include preinstalled Ignition Gateways).

Enabling Hosted Launchers

The following parameter (and value) enables the use of hosted launchers. While enabled, the Gateway will only ever attempt to retrieve the hosted launchers, meaning it will ignore the local launcher files.

Example

```
wrapper.java.additional.##=-Dignition.hostedLaunchers=true
```

Hosted Launcher Version

By default, enabling Hosted Launchers will cause the Gateway to retrieve launchers version appropriate launchers: 8.0.0 launchers for 8.0.0 Gateways, 8.1.0 launchers for 8.1.0 Gateways, etc.

The parameter below can be used to explicitly state which launcher version to retrieve. Generally, this is not recommended, but can potentially be useful if you're looking for a specific launcher version. It requires that `-Dignition.hostedLaunchers` is set to "true".

Example

```
wrapper.java.additional.##=-Dignition.hostedLauncherVersion=8.1.0
```

Ignition Edition

You can specify which edition of Ignition a Gateway should be set to with the parameter demonstrated below.

Example

```
wrapper.java.additional.##=-Dedition=standard
```

Caution: Ignition Gateway Licenses are matched to a specific edition. Changing the edition of an Ignition Gateway that is already licensed can result in the license becoming invalidated. It's recommended that you deactivate a license on a gateway before changing its edition.

Values

- Standard Ignition - standard
- Ignition Edge - edge
- Ignition Maker - maker

This feature is new in Ignition version **8.1.2**
[Click here](#) to check out the other new features

Tag Historian

Tag Historian Query Syntax

The Ignition Tag Historian stores and queries data in a particular way. When a Tag has [Tag History](#) enabled on it, an entry is generated for this Tag inside the `sqlth_te` table in your database. This Tag's Tag path, along with other Tag attributes, are stored in this table. Every Tag entry in the `sqlth_te` table has a unique Tag id which is used by the Tag Historian to identify each Tag. Whenever we make any type of change to this Tag's historical configurations, such as its Tag Group for instance, the current `sqlth_te` table entry for this Tag becomes retired and a new, unretired entry for this Tag gets created with a new Tag id. Due to this dynamic, it is common for a Tag path in the `sqlth_te` table to be associated with more than one Tag id. When a Tag History query executes for a specific Tag, a check is made against the `sqlth_te` table for all the Tag id's that match this Tag's Tag path. These Tag id's are then used to build a dynamic SQL query like the one shown below:

```
SELECT "tagid", "intvalue", "floatvalue", "stringvalue", "datevalue", "t_stamp", "dataintegrity"
FROM sqlth_1_data
WHERE "t_stamp" >= 1580680800000 AND "t_stamp" <= 1581026400000
      AND ( "tagid" = 14568 OR "tagid" = 14571 OR "tagid" = 14572 )
ORDER BY "t_stamp" ASC, "tagid" ASC
```

When we query Tag history for a Tag with three Tag id's associated with its Tag path, the system uses repetitive OR clauses to account for all the Tag id's for this Tag. Additional OR clauses in this fashion can be hard for databases to optimize. For this reason, we changed the query generating mechanism to use the IN operator like so:

```
SELECT "tagid", "intvalue", "floatvalue", "stringvalue", "datevalue", "t_stamp", "dataintegrity"
FROM sqlth_1_data
WHERE "t_stamp" >= 1580680800000 AND "t_stamp" <= 1581026400000
      AND "tagid" IN (14568, 14571, 14572)
ORDER BY "t_stamp" ASC, "tagid" ASC
```

The use of the IN operator allows for better database side query optimization. Users who want to change their historian to use the old query constructor with OR operators can do so by placing the following system flag in the ignition.conf file:

```
wrapper.java.additional.x=-Dignition.taghistorian.useLegacyQuerySyntax=true
```

This feature is new in Ignition version **8.1.2**
[Click here](#) to check out the other new features

Store and Forward

It is possible to adjust the refresh rate of the System Tag Provider's System Tags. To do so, add the following system flag in the ignition.conf file:

```
-DStoreAndForwardStatusTagRefreshRate=####
```

Where "####" is the rate in milliseconds that the Tags should refresh at. Default refresh rate is 5000 milliseconds.

Gateway Port Reference

Port Reference

An Ignition server uses many ports to manage information between itself and other Ignition servers, OPC servers, devices, and other services such as SMTP and ERP servers. The following table describes the default ports an Ignition server may use for communication. Most are configurable and some of these ports may be on a computer other than the one Ignition is installed on (that is, if MySQL is installed on a different computer, the Ignition server will not be using port 3306). This is not a complete list of TCP ports, and there are a few UDP and broadcast ports in the list. Ignition will potentially use other ports if you are connecting to additional services or devices. Setting ports may be possible for some of the following, but will depend on their respective software.

On this page ...

- [Port Reference](#)

Ignition Ports				
Port	Direction	Protocol	Configurable	Description
8088	Incoming	tcp	Yes	Default port setting to access the Ignition Gateway . Also the non SSL port for the Gateway Network.
8043	Incoming	tcp	Yes	Default SSL port setting to access the Ignition Gateway
8060	Incoming	tcp	Yes	Default SSL port setting for the Gateway Network (metroSSLPot in the Gateway XML file)
62541	Incoming	tcp	Yes	Default port for Ignition OPC UA server.
4096	Incoming	tcp	Yes	Legacy port for Ignition OPC UA server. Versions 7.9 and prior used this port for UA server communications. When upgrading from Ignition 7 to 8+, the legacy port will initially be used.
4446	Incoming	udp/broadcast	Yes	Default receive port for a multicast messages that makes the Gateway discoverable on a local network.
6501	Incoming	tcp	No	Server fallback port, used for Local Client Fallback .
17342	Incoming	tcp	No	Receive Port for SMS with Alarming
4445	Outgoing	udp/broadcast	Yes	Default send port for a multicast messaged that makes the Gateway discoverable on a local network.
17341	Outgoing	tcp	No	Send port for SMS with Alarming
1883	Incoming	tcp	No	MQTT Unencrypted. Outgoing port if using an external MQTT broker, Listen port on Ignition if using the Distributor module
8883	Incoming	tcp	No	MQTT SSL/TLS. Outgoing port if using an external MQTT broker, Listen port on Ignition if using the Distributor module
47808	Incoming + Outgoing	udp	Yes	Default BACnet Driver communication port.

Other Common Software / Hardware Ports

Port	Operation	Protocol	Configurable	Description
102	Outgoing	tcp	No	Siemens Step7
2222	Outgoing	tcp	No	Allen Bradley Drivers (Ethernet/IP I/O DMA)
5060	Incoming	tcp/SIP	No	Send Voice Notification to SIP server
5500	Incoming	tcp	Yes	Default port for OPC browse of external Tags
6501	Incoming	tcp	No	Server fallback port
8000	Incoming	tcp/RTP	No	Transfer/com for SIP server
9600	Outgoing	tcp	No	Omron FINS
9600	Incoming + Outgoing	udp	No	Omron FINS
44818	Outgoing	tcp	No	Allen Bradley Drivers (Ethernet/IP Symbolic/General)
47808	Incoming + Outgoing	udp	No	BACnet
135	Outgoing	tcp	No	Default port for DCOM communication (old OPCDA servers)
389	Outgoing	tcp	Yes	Default port for Active directory if this is being used

465	Outgoing	tcp	No	SMTP protocol used if Alarming is configured
502	Outgoing	tcp	Yes	Default Modbus port
1433	Outgoing	tcp	Yes	Default MSSQL port used for connection
1521	Outgoing	tcp	Yes	Default Oracle port used for connection
3050	Outgoing	tcp	Yes	Default Firebirdsql port used for connection
3306	Outgoing	tcp	Yes	Default MySQL port used for connection
5432	Outgoing	tcp	Yes	Default Postgresql port used for connection
50000	Outgoing	tcp	Yes	Default IBM DB2 port used for connection
49320	Outgoing	tcp	Yes	Default Kepware port used for connection

Ignition Auto-Generated SQL Tables

Ignition has a lot of systems built in that will query the database automatically without requiring you to build a query. These systems automatically create the necessary tables in the database, insert the relevant data, and even query it back out for you. However, because this data is all stored in simple database tables, it can be manually accessed using queries to customize how you see the data.

Caution: These tables are setup in very specific ways that the system understands. Altering them in any way may make it so the system can no longer automatically interact with the tables. While it can be useful to manually query out the data, we advise caution when attempting to alter the data or tables in any way. We recommend taking a backup of the database tables before making manual changes to them, with the understand that manually altering the data or tables is done at your own risk.

On this page ...

- [Tag History](#)
 - [Table Structure](#)
 - [sqlt_data_X_X_X](#)
 - [sqlth_drv](#)
 - [sqlth_partitions](#)
 - [sqlth_sce](#)
 - [sqlth_scinfo](#)
 - [sqlth_te](#)
 - [sqlth_annotations](#)
- [Alarm Journal](#)
- [Authentication](#)
- [Audit Log](#)

Tag History

The [Tag History](#) system utilizes six different tables in the database:

Table Name	Table Description	Column References
sqlt_data_x_x_x	This table stores the raw Tag data. There will be multiple tables that fit this format depending on the name of the Gateway and the date. (sqlt_data_1_2018_01 This table is storing data from the Gateway with an id of 1, for the year 2018, for the month of January)	sqlt_data_x_x_x.tagid = sqlth_te.id
sqlth_te	This table stores the non-data details of each Tag.	sqlth_te.scid = sqlth_scinfo.id
sqlth_scinfo	This table stores scan class information.	sqlth_scinfo.drvid = sqlth_drv.id
sqlth_sce	This table stores start and end times for scan classes.	sqlth_sce.scid = sqlth_scinfo.id
sqlth_partitions	This table stores start and end times for each sqlt_data table.	sqlth_partitions.drvid = sqlth_drv.id
sqlth_drv	This table stores information about the drivers of the historical data.	none
sqlth_annotations	This table stores annotations for the Tag history system, such as those created by the Power Chart	none

Table Structure

- [sqlt_data_x_x_x](#): This table stores the raw Tag data. There will be multiple tables that fit this format with date information where any "x" is.
- [sqlth_drv](#): This table stores information about the drivers.
- [sqlth_partitions](#): This table stores start and end times for each sqlt_data table.
- [sqlth_sce](#): This table stores start and end times for scan classes.
- [sqlth_scinfo](#): This table stores scan class information.
- [sqlth_te](#): This table stores non-data details about each Tag.
- [sqlth_annotations](#): This table stores annotations.

sqlt_data_X_X_X

This is the central table that stores the core Tag values. The system stores data in the corresponding xxxvalue column and leaves the others set to NULL. By default, the partition size is set to 1 month and the table name will appear as sqlt_data_1_yyyy_mm. When you first start storing historical data with this system, there will be only one table.

Column Name	Data Type	Notes
dataintegrity	int	Quality of the Tag for this timestamp. 192 is Good Quality, anything else is bad. See Tag Quality Overlays .
datevalue	date	Holds the value of the Tag if it is data type 3, NULL otherwise.
floatvalue	double	Holds the value of the Tag if it is data type 1, NULL otherwise.
intvalue	int	Holds the value of the Tag if it is data type 0, NULL otherwise.
stringvalue	string	Holds the value of the Tag if it is data type 2, NULL otherwise.
t_stamp	long	Unix Timestamp (milliseconds since epoch) for this value.
tagid	int	Unique id of the Tag. References sqlth_te.

sqlth_drv

Column Name	Data Type	Notes
id	int	Unique id of the driver.
name	string	Name of the driver. This is usually the project name.
provider	string	Name of the Tag provider.

sqlth_partitions

This table defines the "partitions" (tables) that are used to store data, and what time frames they cover. Partitioning in the history system splits data across multiple tables in a way that is compatible with all database systems, making certain maintenance tasks easier. The query system does not expect any particular partition configuration, it simply consults this table for table-to-time associations, and then queries the resulting tables for data.

Column Name	Data Type	Notes
pname	string	The name of the table that contains this partition's data.
drv_id	int	The id of the driver that owns this data table. Partitions are created per driver to keep data separate. References sqlth_drv.
start_time	long	Unix Timestamp (milliseconds since epoch) for the earliest time covered by this partition.
end_time	long	Unix Timestamp (milliseconds since epoch) for the end time covered by this partition.
blocksize	int	The size, in milliseconds, of time covered by each entry. This is used by "pre-processed" partitions and would be 0 for normal data partitions.
flags	int	Additional flags that affect how the partition is used. 1 = No seed query support. The system will not execute "bounding value" (or "seed") queries against the table. Useful for database engines that do not support indexing (such as MySQL Archive engine), as these operations can become very time intensive.

sqlth_sce

Column Name	Data Types	Notes
scid	int	Id of the scan class execution entry. References sqlth_scinfo
start_time	long	Unix Timestamp (milliseconds since epoch) for the first execution of this scan class.
end_time	long	Unix Timestamp (milliseconds since epoch) for the latest execution of this scan class.
rate	int	The rate (in milliseconds) of execution. There are no entries for the 'Execute on Value Change' option.

sqlth_scinfo

Column Name	Data Type	Notes
id	int	Unique id of the Scan class.
scname	string	Name of the scan class. "_exempt_" for 'Execute on Value Change' option.
drv_id	int	The driver this scan class uses. References sqith_drv.

sqlth_te

Column Name	Data Type	Notes
id	int	Unique id of the Tag.
tagpath	string	Path of the Tag in the Tag Provider, i.e., Folder1/tag1.
scid	int	The scan class this Tag is storing values with. References sqith_scinfo.
data type	int	The type of value for this Tag. 0: int, byte, short, boolean 1: float, double, long 2: string 3: date
querymode	int	Which internal mode to use for returning data.
created	long	Unix Timestamp (milliseconds since epoch) for when the Tag was created.
retired	long	Unix Timestamp (milliseconds since epoch) for when the Tag was retired (deleted or renamed). This value is NULL while the Tag is active.

sqlth_annotations

This table stores annotations created by the user. Introduced in 8.1.0.

Column Name	Data Type	Notes
id	int	Unique id of the Tag.
tagid	int	The Tag id that the annotation pertains to. References the id values on the sqlth_te table.
start_time	long	The starting point for the annotation.
end_time	long	The ending point for the annotation.
type	string	Represents the type of annotation. Currently the only defined type is "note", which represents a string that corresponds to a particular point of data.
datavalue	string	The value associated with the annotation. When "type" is set to "note", this column represents the content of the user created note.

Alarm Journal

The [Alarm Journal](#) system utilizes two different tables in the database:

Table Name	Table Description	Column References
alarm_events*	This table stores every event (active, cleared, acknowledged) that happened to any alarms that fit within the Journal filter parameters. Each row is a new event	alarm_events.id = alarm_event_data.id
alarm_event_data*	This table stores unique information pertaining to each event. Each row is a specific property of a specific event, so alarm events with multiple properties will have multiple rows in the table.	none

See the [Journal Properties and Tables](#) page for more information regarding all of the columns in the tables.



The names of the tables are completely configurable in the Journal settings in the Gateway. The default table names are used in the table.

Authentication

The [Database Authentication](#) system utilizes six different tables in the database:

Table Name	Table Description	Column References
scada_users*	This table stores each user contained within the user source, along with basic user information. Each row is a new user.	none
scada_roles*	This table stores all of the possible roles within the user source. Each row is a new role.	none
scada_user_rl*	This table stores a mapping of users to roles. Each row is a user and a paired role, so users with multiple roles will have multiple rows in the table.	scada_users_rl.user_id = scada_users.id scada_users_rl.role_id = scada_roles.id
scada_user_sa*	This table stores a list of all upcoming schedule adjustments for each user. Each row is a new schedule adjustment, so users with multiple schedule adjustments will have multiple rows in the table.	scada_user_sa.user_id = scada_users.id
scada_user_ci*	This table stores a list of all contact information items for each user. Each row is a new contact information item, so users with multiple contact information items will have multiple rows in the table.	scada_user_ci.user_id = scada_users.id
scada_user_ex*	This table stores a list of all extra properties for each user, with properties and values stored 1 for 1. Each row is a new property and value pair, so users with multiple extra properties will have multiple rows in the table. Extra properties are added by modules that want to associate data with a user, such as the Voice Notification Module, which adds a Security PIN setting.	scada_user_ex.user_id = scada_users.id

The prefix of the tables are configurable in the User Source settings in the Gateway. The default prefix of "scada_" is used in the table

Audit Log

The [Audit](#) system utilizes one table in the database:

Table Name	Table Description	Column References
AUDIT_EVENTS*	This table stores each auditable event (save, publish, edits, etc.) that has happened for each project or system that has auditing enabled. Each row is a new event.	none

The names of the tables are completely configurable in the Audit settings in the Gateway. The default table names are used in the table.

Related Topics ...

- [Tag Historian](#)
- [Audit Log and Profiles](#)
- [Database Authentication](#)
- [Alarm Journal](#)

Perspective Event Types Reference

Perspective Events provide the opportunity to handle any type of user input or session event as you see fit. To this end, Perspective exposes dozens of configurable events in the **Configure Events...** dialog. The goal of this page is to explain what circumstances trigger each event.

If you choose to associate a given event with a script action, you will be provided with an **event** parameter in your script. List the properties of that **event** parameter, which differs for each broad class of event.

Component Events

Component Events serve the same purpose as Vision extension functions, and are visible and configurable only on components for which they are applicable. The following components have component events:

- [Table Component](#)
- [File Upload Component](#)
- [View Canvas](#)
- [Signature Pad](#)

Refer to the respective component pages for descriptions of component events.

Action Performed Event

The action performed event is on many components where an action of some sort may occur. While the action may differ between components, the event functions in the same way.

Applicable components:

- Date Time Input
- Date Time Picker
- Barcode Scanner Input
- Button
- Checkbox
- Dropdown
- Multi-State Button
- Numeric Entry Field
- One-Shot Button
- Radio Group
- Slider
- Toggle Switch

Event Name	Description
onActionPerformed	Fired when the action of the component is done, such as the button being pressed, or a checkbox being checked.

System Events

System Events are events that are not related to any specific user input, but to status changes of the *component* or *view*.

Event Name	React Event	Description
onStartup	componentDidMount	<ul style="list-style-type: none">• Fired on a component when the component's view is loaded into the browser window.• Fired on a view when the view is loaded into the browser window.
onShutdown	componentWillUnmount	<ul style="list-style-type: none">• Fired on a component when the component's view is removed from the browser window (or when the session closes due to logout or timeout)• Fired on a view when the view is removed from the browser window (or when the session closes due to logout or timeout)

On this page ...

- [Component Events](#)
 - [Action Performed Event](#)
- [System Events](#)
 - [Selection Event Properties](#)
- [Mouse Events](#)
 - [Mouse Event Properties](#)
- [Keyboard Events](#)
 - [Keyboard Event Properties](#)
 - [Selection Event Properties](#)
- [Focus Events](#)
 - [Selection Event Properties](#)
- [Selection Events](#)
 - [Selection Event Properties](#)
- [Touch Events](#)
 - [Touch Event Properties](#)
- [Wheel Events](#)
 - [Wheel Event Properties](#)

Selection Event Properties

There are no special properties associated with a selection event.

Mouse Events

Event Name	Description
onClick	Fired when a mouse button is pressed <i>and</i> released over the element with the event configured.
onContextMenu	Fired when the <i>right</i> mouse button is pressed and released over the element.
onDoubleClick	Fired when a mouse button is pressed and released twice over the element.
onMouseDown	Fired when a mouse button is pressed (<i>not necessarily released</i>) over the element.
onMouseEnter	Fired when the mouse pointer is moved onto the element.
onMouseLeave	Fired when the mouse pointer is moved off the element.
onMouseMove	Fired continuously while the mouse moves over the element.
onMouseOut	Fired when the mouse pointer is moved off the element.
onMouseOver	Fired when the mouse pointer is moved onto the element.
onMouseUp	Fired when a mouse button is released (<i>not necessarily pressed</i>) over the element.

Mouse Event Properties

Property Name	Description
altKey	Is <i>true</i> if the Alt key is pressed when the event is fired, and <i>false</i> otherwise.
button	Indicates which mouse button was used to perform the click event. Possible options are: <ul style="list-style-type: none">• 0 - Main button (usually the left mouse button)• 1 - Auxiliary button (usually the wheel button or middle mouse button)• 2 - Secondary button (usually the right mouse button)• 3 - Fourth button (usually the <i>Browser Back</i> button)• 4 - Fifth button (usually the <i>Browser Forward</i> button)
buttons	For handling situations where multiple mouse buttons are pressed when the event is fired, <i>buttons</i> holds an integer sum, where each button is given the following values: <ul style="list-style-type: none">• 1 - Main button• 2 - Secondary button• 4 - Auxiliary button• 8 - Fourth button• 16 - Fifth button This, if the main and fourth buttons are pressed simultaneously, the <i>buttons</i> value will be $1 + 8 = 9$.
clientX	Indicates the horizontal coordinate of the mouse click relative to the edge of the session.
clientY	Indicates the vertical coordinate of the mouse click relative to the edge of the session.
ctrlKey	Is <i>true</i> if the Ctrl key is pressed when the event is fired, and <i>false</i> otherwise.
getModifierState(key)	Returns the current state of the specified modifier key: <i>true</i> if the modifier is active (i.e., the modifier key is pressed or locked), otherwise, <i>false</i> .
metaKey	Is <i>true</i> if the meta key (like the <i>Windows</i> key in Windows) is pressed when the event is fired, and <i>false</i> otherwise.
pageX	Same as <i>clientX</i> , except horizontal scrolling is taken into account. If some of the session is hidden on the left side, <i>pageX</i> will be larger than <i>clientX</i> .
pageY	Same as <i>clientY</i> , except vertical scrolling is taken into account. If some of the session is hidden above, <i>pageY</i> will be larger than <i>clientY</i> .
relatedTarget	A <i>read-only</i> property that is the secondary target for the mouse event, if there is one. For events with no secondary target, <i>relatedTarget</i> returns <i>null</i> .

screenX	Provides the horizontal coordinate (offset) of the mouse pointer in global (screen) coordinates.
screenY	Provides the vertical coordinate (offset) of the mouse pointer in global (screen) coordinates.
shiftKey	Is <i>true</i> if the Shift key is pressed when the event is fired, and <i>false</i> otherwise.

Keyboard Events

Event Name	Description
onKeyDown	Fired whenever any key is <i>pressed</i> while the element is focused.
onKeyPress	Fired continuously while any key (other than Shift, Fn, and CapsLock) is in a pressed position and the element is focused. This event isn't recommended. Use onKeyDown instead. MDN keypress event
onKeyUp	Fired whenever any key is <i>released</i> while the element is focused.

Keyboard Event Properties

Property Name	Description
altKey	Is <i>true</i> if the Alt key is pressed when the event is fired; <i>false</i> otherwise.
charCode	Read-only property that returns the Unicode value of a character key pressed during a keypress event. Deprecated. Use the <code>key</code> property instead: MDN charCode
ctrlKey	Is <i>true</i> if the Ctrl key is pressed when the event is fired; <i>false</i> otherwise.
getModifierState(key)	Returns the current state of the specified modifier key: <i>true</i> if the modifier is active (i.e., the modifier key is pressed or locked); <i>false</i> otherwise.
key	Read-only property that returns the value of the key pressed by the user, taking into consideration the state of modifier keys such as Shift as well as the keyboard locale and layout.
keyCode	Read-only property returning a system and implementation dependent numerical code, identifying the unmodified value of the pressed key. Deprecated: MDN keyCode
location	Read-only property that returns a value representing the location of the key on the keyboard or other input device.
metaKey	Is <i>true</i> if the meta key (like the Windows key in Windows) is pressed when the event is fired; <i>false</i> otherwise.
repeat	Is <i>true</i> if the given key is being held down such that it is automatically repeating.
shiftKey	Is <i>true</i> if the Shift key is pressed when the event is fired; <i>false</i> otherwise.
which	Read-only property that returns the numeric <code>keyCode</code> of the key pressed, or the character code (<code>charCode</code>) for an alphanumeric key pressed. Deprecated: MDN which

Composition Events

The following events are triggered via [composition](#) - meaning cases where supplementary or alternative means are used to input text. For example, building logograms from base characters, or when voice commands are converted into text via speech recognition.

Event Name	Description
onCompositionEnd	Triggers when composition ends on an input component.
onCompositionStart	Triggers when composition begins on an input component.
onCompositionUpdate	Triggers when a new character is received while composing on an input component.

Selection Event Properties

There are no special properties associated with a selection event.

Focus Events

Event Name	Description
onFocus	Fired whenever the element is focused. Usually, a component or container becomes focused when the user starts to interact with it.
onBlur	Fired when the element loses focus. This usually happens when the user focuses something else.

Selection Event Properties

There are no special properties associated with a selection event.

Selection Events

Event Name	Description
onSelect	Fired after a section of text is selected in the element.

Selection Event Properties

There are no special properties associated with a selection event.

Touch Events

Event Name	Description
onTouch Cancel	Fired when a touch point has been disrupted, such as when a touch manipulation (i.e., panning or zooming) is initiated or more touch points are added to the surface than the device is capable of registering.
onTouch End	Fired when the user removes a touch point from the touch surface, including cases where the touch point physically leaves the touch surface, such as being dragged off of the screen.
onTouch Move	Fired when a touch point has moved along the touch surface.
onTouch Start	Fired when the user has touched the surface of a touch capable device.

Touch Event Properties

Property Name	Description
altKey	Is <i>true</i> if the Alt key is pressed when the event is fired; <i>false</i> otherwise.
ctrlKey	Is <i>true</i> if the Ctrl key is pressed when the event is fired, and <i>false</i> otherwise.
metaKey	Is <i>true</i> if the meta key (like the <i>Windows</i> key in Windows) is pressed when the event is fired; <i>false</i> otherwise.
shiftKey	Is <i>true</i> if the Shift key is pressed when the event is fired; <i>false</i> otherwise.

Wheel Events

Event Name	Description
onWheel	Fired whenever the mouse's scroll wheel is used over the element.

Wheel Event Properties

--	--

Property Name	Description
deltaMode	Indicates the <i>unit</i> associated with the other three properties. Possible values are: <ul style="list-style-type: none"> • 0 - The other properties specify the wheel movement in pixels. • 1 - The other properties specify the wheel movement in lines. • 2 - The other properties specify the wheel movement in pages.
deltaX	Indicates the length of a horizontal scroll. Most mouse devices do not possess horizontal scrolling capability. Scrolling right will return a positive value, and scrolling left will return a negative value.
deltaY	Indicates the length of a vertical scroll. This is by far the most common type of scroll; it coincides with the upward and downward movement of a typical mouse scroll wheel. deltaY will be positive when scrolling down, and negative when scrolling up.
deltaZ	Indicates the length of a <i>depth</i> scroll. Not many mouse devices deliver this kind of scroll; typically only a "3D" mouse would control this scroll type. The value is positive when scrolling in, and negative when scrolling out.

Related Topics ...

- [Alarm Associated Data](#)
- [Alarm Notification Pipelines](#)
- [Tag Properties](#)
- [Component Events and Actions](#)

Style Reference

The following table describes the style user interface for style properties in Perspective. These styles are based on Cascading Style Sheets (CSS), a style sheet language used for describing the presentation of a document or webpage. For an overview on using styles in Perspective, see also [Styles](#) and [Style Classes](#).

On this page ...

- [Text Menu](#)
- [Background Menu](#)
- [Margin and Padding Menu](#)
- [Border Menu](#)
- [Shape Menu](#)
- [Misc Menu](#)

Text Menu

Text

Font family **Size**

Color **Weight**

Line height **Letter spacing** **Word spacing**

Alignment

Text align

Indent **White space**

Text Options

Transform **Decoration**

Tt TT tt S ~~S~~ —S Break-word

Shadow

Color **X-Offset** **Y-Offset** **Blur**

Advanced ▲

Overflow wrap

normal break-word

Text overflow

clip ellipsis

Property Name	Description	Example	Output
Font Family	Specifies the font for the component. Examples of a Font Family are Times, Courier, Arial, Monospace, etc.	fontFamily: Monospace	Hello, This is a monospace font family.
Size	Specifies the size of the font in pixels (px) or points (pt). If you enter just a number, Perspective assumes	fontSize: 26 px	

	the value is in pixels.		Hello												
Color	Specifies the color of the text. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .	color: #FF0000	Hello												
Weight	Sets how thick or thin characters in the text are displayed.	fontWeight: Bold fontSize: 26px	Hello												
	<table border="1"> <thead> <tr> <th>Style</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>normal</td> <td>Defines normal characters. This is the default.</td> </tr> <tr> <td>bold</td> <td>Defines thick characters.</td> </tr> <tr> <td>lighter</td> <td>Defines lighter characters.</td> </tr> <tr> <td>bolder</td> <td>Defines thicker characters.</td> </tr> <tr> <td>100-900</td> <td>Defines from thin to thick characters. 400 is the same as normal. 700 is the same as bold.</td> </tr> </tbody> </table>	Style	Description	normal	Defines normal characters. This is the default.	bold	Defines thick characters.	lighter	Defines lighter characters.	bolder	Defines thicker characters.	100-900	Defines from thin to thick characters. 400 is the same as normal. 700 is the same as bold.	fontWeight: Lighter fontSize: 26px	Hello
Style	Description														
normal	Defines normal characters. This is the default.														
bold	Defines thick characters.														
lighter	Defines lighter characters.														
bolder	Defines thicker characters.														
100-900	Defines from thin to thick characters. 400 is the same as normal. 700 is the same as bold.														
Italic	Checkbox to indicate if the font style should be italic.	fontStyle: italic	<i>Hello</i>												
Line Height	Specifies the height of a line. Can be specified as one of the following:	lineHeight: normal lineHeight: 32px	Hello World! Hello World! Hello World! Hello World!												
Letter Spacing	Increases or decreases the space between characters. The default is 0.	letterSpacing: 12px	H e l l o W o r l d												
Word Spacing	Increases or decreases the white space between words. The default is normal.	wordSpacing: 24px	Hello World												
Alignment															
Text Align	The horizontal alignment of text in a component.	textAlign: right textAlign: center textAlign: justify	<div style="border: 1px solid black; padding: 5px; text-align: right;">Hello</div> <div style="border: 1px solid black; padding: 5px; text-align: center;">Hello</div> <div style="border: 1px solid black; padding: 5px; text-align: justify; width: 300px;">Welcome to Ignition by Inductive Automation. We are pleased that you are using our SCADA software.</div>												
Text Indent	The indentation of the first line in a text-block. The default value is 0.	textIndent: 24px													

			<p>Welcome to Ignition by Inductive Automation. We are pleased that you are using our SCADA software.</p>												
White space	<p>Specifies how white-space inside a component is handled.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>normal</td><td>Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is the default.</td></tr> <tr> <td>nowrap</td><td>Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a
 tag is encountered.</td></tr> <tr> <td>pre</td><td>Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <pre> tag in HTML.</td></tr> <tr> <td>pre-wrap</td><td>Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary, and on line breaks.</td></tr> <tr> <td>pre-line</td><td>Whitespace is preserved by the browser. Text will wrap when necessary, and on line breaks.</td></tr> </tbody> </table>	Style	Description	normal	Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is the default.	nowrap	Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a tag is encountered.	pre	Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <pre> tag in HTML.	pre-wrap	Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary, and on line breaks.	pre-line	Whitespace is preserved by the browser. Text will wrap when necessary, and on line breaks.	whiteSpace: pre-wrap	<p>Welcome to Ignition by Inductive Automation. We are pleased that you are using our SCADA software.</p>
Style	Description														
normal	Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is the default.														
nowrap	Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a tag is encountered.														
pre	Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <pre> tag in HTML.														
pre-wrap	Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary, and on line breaks.														
pre-line	Whitespace is preserved by the browser. Text will wrap when necessary, and on line breaks.														

Text Options

Transform	<p>Capitalization setting for the text.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Icon</th><th style="text-align: left;">Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Tt</td><td>capitalize</td><td>Transforms the first character of each word to uppercase.</td></tr> <tr> <td>TT</td><td>uppercase</td><td>Transforms all characters to uppercase.</td></tr> <tr> <td>tt</td><td>lowercase</td><td>Transforms all characters to lowercase.</td></tr> </tbody> </table>	Icon	Style	Description	Tt	capitalize	Transforms the first character of each word to uppercase.	TT	uppercase	Transforms all characters to uppercase.	tt	lowercase	Transforms all characters to lowercase.	textTransform: uppercase	<p>HELLO</p>
Icon	Style	Description													
Tt	capitalize	Transforms the first character of each word to uppercase.													
TT	uppercase	Transforms all characters to uppercase.													
tt	lowercase	Transforms all characters to lowercase.													
Decoration	<p>Text decoration setting.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Icon</th><th style="text-align: left;">Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>S</td><td>underline</td><td>Horizontal line under text.</td></tr> <tr> <td>S</td><td>line-through</td><td>Horizontal line through text.</td></tr> <tr> <td>S</td><td>overline</td><td>Horizontal line above text.</td></tr> </tbody> </table>	Icon	Style	Description	S	underline	Horizontal line under text.	S	line-through	Horizontal line through text.	S	overline	Horizontal line above text.	textDecoration: line-through	<p>Hello</p>
Icon	Style	Description													
S	underline	Horizontal line under text.													
S	line-through	Horizontal line through text.													
S	overline	Horizontal line above text.													
Break-word	<p>Allows unbreakable words to be broken.</p>	wordWrap: break-word													

			<p>Hello! Welcome to Ignition by Inductive Automati on. We</p>											
Shadow	Adds shadow to text. Shadow is set in pixels.	<table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Color</td><td>Color for the shadow. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference.</td></tr> <tr> <td>x-offset</td><td>Offset distance on x axis, in pixels</td></tr> <tr> <td>y-offset</td><td>Offset distance on y axis, in pixels</td></tr> <tr> <td>blur-radius</td><td>The blur-radius. The default value is 0.</td></tr> </tbody> </table>	Style	Description	Color	Color for the shadow. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .	x-offset	Offset distance on x axis, in pixels	y-offset	Offset distance on y axis, in pixels	blur-radius	The blur-radius. The default value is 0.	textShadow: #ff0000 6px 6px 2px	<p>Hello Hello</p>
Style	Description													
Color	Color for the shadow. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .													
x-offset	Offset distance on x axis, in pixels													
y-offset	Offset distance on y axis, in pixels													
blur-radius	The blur-radius. The default value is 0.													
Overflow wrap	Allows the browser to break a line of text inside a component into multiple lines in an otherwise unbreakable place, thus avoiding a long string of text causing layout problems due to overflow.	<table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>normal</td><td>Breaks lines according to normal line breaking rules. Words or unbroken strings will not break, even if they overflow the container. This is the default.</td></tr> <tr> <td>break-word</td><td>Words or strings of characters that are too large to fit inside their component will break in an arbitrary place to force a line break. A hyphen character will not be inserted, even if the hyphen property is used.</td></tr> </tbody> </table>	Style	Description	normal	Breaks lines according to normal line breaking rules. Words or unbroken strings will not break, even if they overflow the container. This is the default.	break-word	Words or strings of characters that are too large to fit inside their component will break in an arbitrary place to force a line break. A hyphen character will not be inserted, even if the hyphen property is used.	overflow wrap: normal	<p>Hello World. Demo overfl ow wrap normal property</p>				
Style	Description													
normal	Breaks lines according to normal line breaking rules. Words or unbroken strings will not break, even if they overflow the container. This is the default.													
break-word	Words or strings of characters that are too large to fit inside their component will break in an arbitrary place to force a line break. A hyphen character will not be inserted, even if the hyphen property is used.													
Text overflow	Specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped or it can display an ellipsis (...).	<table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>clip</td><td>The text is clipped and not accessible. This is the default.</td></tr> <tr> <td>ellipsis</td><td>Render an ellipsis ("...") to represent the clipped text.</td></tr> </tbody> </table>	Style	Description	clip	The text is clipped and not accessible. This is the default.	ellipsis	Render an ellipsis ("...") to represent the clipped text.	textOverflo w: clip	<p>Hello World. Demo textOverflow clip styl</p>				
Style	Description													
clip	The text is clipped and not accessible. This is the default.													
ellipsis	Render an ellipsis ("...") to represent the clipped text.													
			textOverflo w: ellipsis	<p>Hello World. Demo textOverflow ellipsis styl...</p>										

Background Menu

Background

Background color

Background image

Background position

Advanced ▾

Background clip

border-box	padding-box	content-box
------------	-------------	-------------

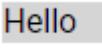
Background repeat

repeat	repeat-x	repeat-y	no-repeat
--------	----------	----------	-----------

Background attachment

scroll	fixed	local
--------	-------	-------

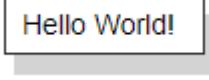
Box shadow

Property Name	Description	Example	Output
Background Color	Sets the background color of a component. The background of a component is the total size of the component, including padding and border (but not the margin). You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .	backgroundColor: #D5D5D5	
Background Image	<p>Sets one or more background images for a component. By default, a background-image is placed at the top-left corner of a component, and repeated both vertically and horizontally.</p> <p>This setting also has options for setting a linear or radial gradient as the background image.</p>	<pre>backgroundImage: url ("http://files.inductiveautomation.com/training/user_manual_images/ignition_logo.png") backgroundRepeat: no-repeat textAlign: center</pre>	 (image resized)
Background Position	Sets the starting position of a background image. By default, a background-image is placed at the top-left corner of a component, and repeated both vertically and horizontally. Values can be as follows:	<pre>backgroundPosition: 100px 100px borderStyle: solid borderWidth: 3 px</pre>	

n gt h in pi x els	The first value is the horizontal position, second value is the vertical position. For example 100px 5px will move the image 100px to the right and five pixels down.	fontSize: 56px backgroundImage: url ("https://cdn.pixabay.com/photo/2017/02/20/18/03/cat-2083492_960_720.jpg")	 (image resized)
p e rc e nt a g e s	Moving a background image by a percentage means it will align the percentage point in the image to the same percentage point in the container. For example, 50% means it would align the middle of the image with the middle of the container., where 100% means it will align the last pixel of the image with the last pixel of the container		
k e y w o r d s	Options are top, bottom, left, right, and center.		

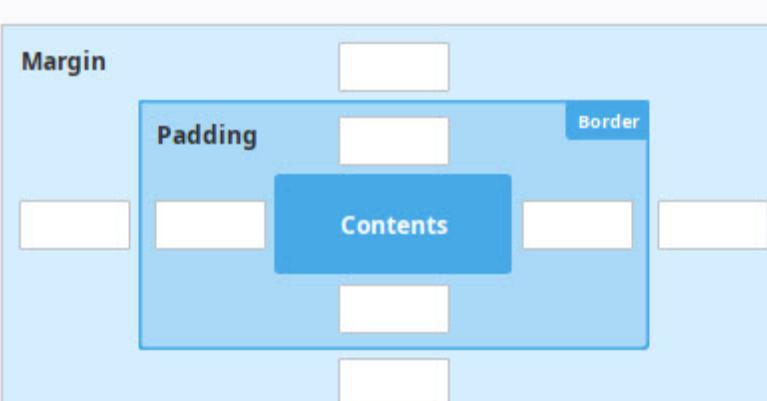
Advanced

Background Clip	Defines how far the background (color or image) should extend within a component. (Be sure to set up the border, padding, background properties, and add some content before configuring the background-clip, otherwise you will not see any changes when you add a value for the backgroundClip.	backgroundClip: content-box backgroundColor: #CCCCFF borderStyle: dotted border-box padding-box content-box	 (image resized)
Background Repeat	Sets if/how a background image will be repeated. By default, a background-image is repeated both vertically and horizontally.	backgroundRepeat: repeat backgroundImage: url ("http://files.inductiveautomation.com/training/user_manual_images/ignition_logo.png") 	 (image resized)
Background Attachment	Specifies whether a background image scrolls with the rest of the page, or is fixed.	backgroundAttachment: fixed	

	contents.		
Box Shadow	Attaches one or more shadows to a component. No shadow is the default.	boxShadow: 5px 10px #D3D3D3	

Margin and Padding Menu

Margin and Padding



The diagram shows a large light blue rectangle representing the 'Margin'. Inside it is a smaller blue rectangle representing the 'Border'. Within the border is a white rectangle representing the 'Padding'. The innermost part, which contains the text 'Contents', is represented by a dark blue rectangle.

Margin	Padding
	

Property Name	Description	Example	Output
Padding Top /Left/Bottom /Right	Padding is the space inside of a component, between the border and the content of the component. This property sets the padding for each of the four sides inside of a component.	paddingBottom: 2 paddingLeft: 30 paddingRight: 13 paddingTop: 13	
Margin Top /Left/Bottom /Right	Margin is the space outside of a component. This property sets the padding for each of the four sides surrounding a component.	marginBottom: marginLeft: marginRight: marginTop:	 marginTop: 10px

Border Menu

Border

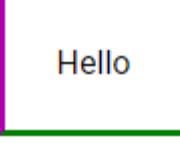
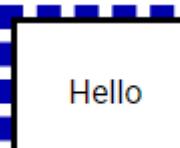
Border

Top style	Top width	Top color
solid	3px	<input type="color"/>
Left style	Left width	Left color
solid	3px	<input type="color"/>
Bottom style	Bottom width	Bottom color
solid	3px	<input type="color"/>
Right style	Right width	Right color
solid	3px	<input type="color"/>
Border radius		
<input type="text"/>	<input checked="" type="checkbox"/> All corners	<input type="text"/>

Outline

Outline style	Outline width	Outline color
<input type="text"/>	<input type="text"/>	<input type="color"/>

Property Name	Description	Example	Output														
Border Unlink Sides / Link Sides	<p>The first item on the Border menu is the Border Unlink sides icon. When you click this icon, the four border sides of the component are "unlinked," and the Border Styles settings can be applied individually to each side.</p> <p>When the Link Sides icon is selected, then the Border Styles settings will apply to <i>all</i> four borders equally.</p>																
Border Style	<p>Sets the style for an component's four borders.</p> <table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr> <td>none</td><td>Specifies no border. This is default.</td></tr> <tr> <td>solid</td><td>Specifies a solid border.</td></tr> <tr> <td>dashed</td><td>Specifies a dashed border.</td></tr> <tr> <td>dotted</td><td>Specifies a dotted border.</td></tr> <tr> <td>double</td><td>Specifies a double border.</td></tr> <tr> <td>groove</td><td>Specifies a 3D grooved border. The effect depends on the border-color value.</td></tr> </tbody> </table>	Style	Description	none	Specifies no border. This is default.	solid	Specifies a solid border.	dashed	Specifies a dashed border.	dotted	Specifies a dotted border.	double	Specifies a double border.	groove	Specifies a 3D grooved border. The effect depends on the border-color value.	borderStyle: dotted	
Style	Description																
none	Specifies no border. This is default.																
solid	Specifies a solid border.																
dashed	Specifies a dashed border.																
dotted	Specifies a dotted border.																
double	Specifies a double border.																
groove	Specifies a 3D grooved border. The effect depends on the border-color value.																

	<table border="1"> <tr><td>ridge</td><td>Specifies a 3D ridged border. The effect depends on the border-color value.</td></tr> <tr><td>inset</td><td>Specifies a 3D inset border. The effect depends on the border-color value.</td></tr> <tr><td>outset</td><td>Specifies a 3D outset border. The effect depends on the border-color value.</td></tr> <tr><td>hidden</td><td>The same as "none", except in border conflict resolution for table elements.</td></tr> </table>	ridge	Specifies a 3D ridged border. The effect depends on the border-color value.	inset	Specifies a 3D inset border. The effect depends on the border-color value.	outset	Specifies a 3D outset border. The effect depends on the border-color value.	hidden	The same as "none", except in border conflict resolution for table elements.		
ridge	Specifies a 3D ridged border. The effect depends on the border-color value.										
inset	Specifies a 3D inset border. The effect depends on the border-color value.										
outset	Specifies a 3D outset border. The effect depends on the border-color value.										
hidden	The same as "none", except in border conflict resolution for table elements.										
Border Width	<p>Sets the width of the border.</p> <p>If the Unlink sides  icon is selected, you can set separate border widths for the top, left, bottom, and right sides.</p>	borderWidth: 7px borderStyle: solid textAlign: center									
		borderTopWidth: 12px borderTopColor: #0000D9 borderTopStyle: solid borderStyle: solid borderWidth: 3px borderColor: #AC00AC textAlign: center									
Border Color	<p>Specifies the color of the border. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference.</p> <p>If the Unlink sides  icon is selected, you can set separate border colors for the top, left, bottom, and right sides.</p>	borderColor: #AC00AC borderWidth: 3px borderStyle: solid alignVertical: center textAlign: center									
		borderTopColor: #4747FF borderLeftColor: #AC00AC borderBottomColor: #008000 borderRightColor: #FF0000 borderWidth: 3px borderStyle: solid									
Border Radius	Defines the radius of the component's corners allowing rounded borders.	borderRadius: 10px borderWidth: 3px borderStyle: solid alignVertical: center textAlign: center									
Border Radii Top/Left /Bottom/Right	Sets the radius for each of the component's corners: Border top left radius, Border top right radius, Border bottom left radius, Border bottom right radius.	borderTopLeftRadius: 10px borderBottomRightRadius: 10px borderStyle: solid									
Outline Style	Specifies the style of an outline.	outlineStyle: dashed outlineWidth: 6px outlineColor: #0000AC borderWidth:									

	dashed	Specifies a dashed outline.	3px borderStyle: solid	
	dotted	Specifies a dotted outline.		
	double	Specifies a double outline.		
	groove	Specifies a 3D grooved outline. The effect depends on the outline-color value.	outlineStyle: double outlineColor: #000AC outlineWidth: 12px borderWidth: 3px borderStyle: solid	
	ridge	Specifies a 3D ridged outline. The effect depends on the outline-color value.		
	inset	Specifies a 3D inset outline. The effect depends on the outline-color value.		
	outset	Specifies a 3D outset outline. The effect depends on the outline-color value.		
	hidden	Specifies a hidden outline.		

Shape Menu



Property Name	Description
Fill	Sets a color for the fill of an SVG shape.
Stroke	Sets a color for the stroke. The stroke is a border around SVG shapes. Specifies the color of the text. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .
Stroke width	Sets the width of the stroke, in pixels.

Misc Menu

▼ Misc.

Opacity

Cursor

Overflow

visible	hidden	scroll	auto
---------	--------	--------	------

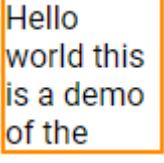
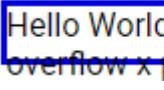
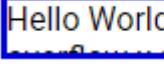
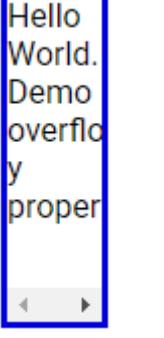
Overflow x

visible	hidden	scroll	auto
---------	--------	--------	------

Overflow y

visible	hidden	scroll	auto
---------	--------	--------	------

Property Name	Description	Example	Output
Opacity	Describes the transparency level for a component, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent. Opacity applies to text, border, and outline properties.	opacity: 0.50	
Cursor	Specifies the mouse cursor to be displayed when pointing over a component.	cursor: pointer cursor: crosshair 	

		<table border="1"> <tr><td></td><td>progress</td></tr> <tr><td></td><td>text</td></tr> <tr><td></td><td>wait</td></tr> <tr><td></td><td>zoom-in</td></tr> </table>		progress		text		wait		zoom-in				
	progress													
	text													
	wait													
	zoom-in													
Overflow	Specifies what should happen if content overflows a component's box: whether to clip content or add scrollbars when a component's content is too big to fit in a specified area.	<table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr><td>visible</td><td>The overflow renders outside the component's box. It is not clipped. This is the default.</td></tr> <tr><td>hidden</td><td>The overflow is clipped, and the rest of the content will be invisible.</td></tr> <tr><td>scroll</td><td>The overflow is clipped, but a scrollbar is added to see the rest of the content.</td></tr> <tr><td>auto</td><td>If overflow is clipped, a scrollbar will be added to see the rest of the content.</td></tr> </tbody> </table>	Style	Description	visible	The overflow renders outside the component's box. It is not clipped. This is the default.	hidden	The overflow is clipped, and the rest of the content will be invisible.	scroll	The overflow is clipped, but a scrollbar is added to see the rest of the content.	auto	If overflow is clipped, a scrollbar will be added to see the rest of the content.	overflow: visible	
Style	Description													
visible	The overflow renders outside the component's box. It is not clipped. This is the default.													
hidden	The overflow is clipped, and the rest of the content will be invisible.													
scroll	The overflow is clipped, but a scrollbar is added to see the rest of the content.													
auto	If overflow is clipped, a scrollbar will be added to see the rest of the content.													
			overflow: hidden											
			overflow: scroll											
Overflow X	Specifies whether to clip the content, add a scroll bar, or display overflow content of a block-level component, when it overflows at the left and right edges.	<table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr><td>visible</td><td>The content is not clipped, and will be rendered outside the left and right edges. This is the default.</td></tr> <tr><td>hidden</td><td>The content is clipped. No scrolling mechanism is provided.</td></tr> <tr><td>scroll</td><td>The content is clipped and a scrolling mechanism is provided.</td></tr> <tr><td>auto</td><td>Will cause a scrolling mechanism to be provided for overflowing boxes.</td></tr> </tbody> </table>	Style	Description	visible	The content is not clipped, and will be rendered outside the left and right edges. This is the default.	hidden	The content is clipped. No scrolling mechanism is provided.	scroll	The content is clipped and a scrolling mechanism is provided.	auto	Will cause a scrolling mechanism to be provided for overflowing boxes.	overflow x: visible	
Style	Description													
visible	The content is not clipped, and will be rendered outside the left and right edges. This is the default.													
hidden	The content is clipped. No scrolling mechanism is provided.													
scroll	The content is clipped and a scrolling mechanism is provided.													
auto	Will cause a scrolling mechanism to be provided for overflowing boxes.													
			overflow x: hidden											
Overflow Y	Specifies whether to clip the content, add a scroll bar, or display overflow content of a block-level component, when it overflows at the top and bottom edges.	<table border="1"> <thead> <tr> <th>Style</th><th>Description</th></tr> </thead> <tbody> <tr><td>visible</td><td>The content is not clipped, and will be rendered outside the content box. This is the default.</td></tr> <tr><td>hidden</td><td>The content is clipped. No scrolling mechanism is provided.</td></tr> <tr><td>scroll</td><td>The content is clipped and a scrolling mechanism is provided.</td></tr> <tr><td>auto</td><td>Will cause a scrolling mechanism to be provided for overflowing boxes.</td></tr> </tbody> </table>	Style	Description	visible	The content is not clipped, and will be rendered outside the content box. This is the default.	hidden	The content is clipped. No scrolling mechanism is provided.	scroll	The content is clipped and a scrolling mechanism is provided.	auto	Will cause a scrolling mechanism to be provided for overflowing boxes.	overflow y: scroll	
Style	Description													
visible	The content is not clipped, and will be rendered outside the content box. This is the default.													
hidden	The content is clipped. No scrolling mechanism is provided.													
scroll	The content is clipped and a scrolling mechanism is provided.													
auto	Will cause a scrolling mechanism to be provided for overflowing boxes.													

Symbol Reference

The following reference pages provide tables showing the symbols and icons used in Ignition along with the symbol name and description.

[Project Browser Symbol Reference](#)

[Tag Browser Symbol Reference](#)

Tag Browser Symbol Reference

This reference section provides tables showing the symbols used in Tag Browser along with the symbol name and description.

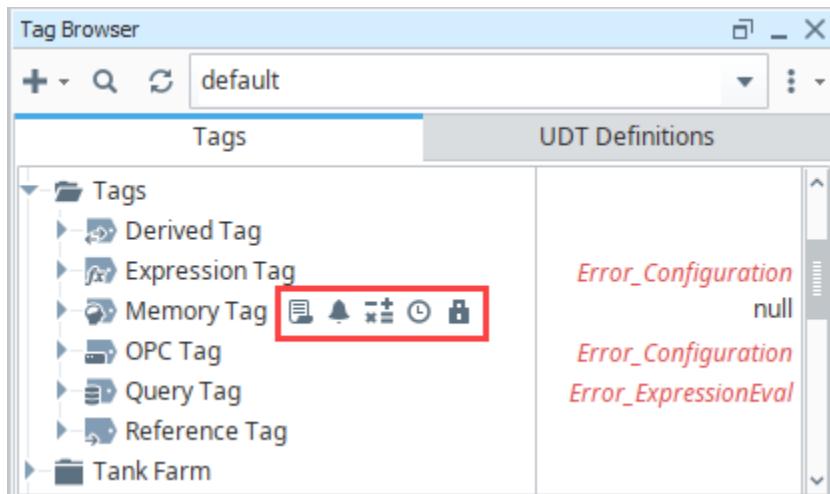
Tag Browser Icons

Symbol	Name	Description
	Add	Add button used in many areas in the Tag Browser such as adding a new Device, Tag Folder, Data Type Instance, and a Tag.
	Column Selection	Options for what columns are displayed in the Tag Browser: Value, Data Type, and Traits. Check the option in the dropdown list to display that column.
	Copy	Copies the selected item, for example, a Tag or a UDT.
	Copy Tag Path	Copies the full tag path to the clipboard.
	Data Types Folder Closed	Icon displayed when the data types folder is closed.
	Data Types Folder Open	Icon displayed when the data types folder is open.
	Data Type Instance	Creates new instance of an existing Data Type. Dropdown list shows the available instances.
	Tag Groups (formerly Scan Classes)	Opens the Tag Group Editor (formerly scan class editor).
	Export Tags	Export Tags.
	Folder Closed	Icon displayed when a folder is closed.
	Folder Open	Icon displayed when a folder is open.
	Import Tags	Opens import window from which you can import a file
	Multi-instance Wizard	Opens the multi-instance wizard .
	New Standard Tag	Dropdown list from which you can select an option to create a new data type, a new data type instance, or a new standard Tag. You can also create a new folder for Tags or data types.
	OPC Server	Browses the OPC servers.
	Paste	Pastes the contents of the clipboard into the current context.
	Paste Cancel	Cancels the paste action.
	Refresh	Refreshes the Tag providers.
	Rename	Enables text editing of the Tag name.
	Search	Search for Tags.
	Tag	Icon for a Tag. Appears next to the Tag name in the Tag Browser tree.

	Tag Disconnected	Indicates the Tag is disconnected.
	Tag Edit	Opens the Tag Editor screen.
	Tree Node	Icon for a node in the tree that's expanded from an individual Tag.

Tag Traits

Tag configurations or settings are visually represented by a unique icon next to the Tag in the Tag Browser.



The following icons enable you to note some important settings on the Tag at a glance. A description of the icons are listed below.

Icon	Setting	Description
	Scaling	The Scale Mode property under the Numeric Tag Properties section of the Tag Editor has been set to a value other than "Off." The value on the Tag will be scaled to some degree.
	Alarming	At least one alarm has been configured for this Tag.
	Tag History	This Tag has been configured to log data into the Tag Historian system.
	Tag Event Script	At least one Tag Event Script has been enabled on this Tag.
	Tag Permissions	The Access Rights property under the Security Tag Properties section of the Tag Editor has been set to Read only or Custom. For more information, see Tag Security Properties .

Project Browser Symbol Reference

This reference section provides tables showing the symbols used in Project Browser along with the symbol name and description.

Project Browser Icons

Symbol	Name	Description
	Alarming	Workspace for Alarm Notification Pipelines . Appears under the Global resources. The same icon is displayed next to the name of each pipeline.
	Sequential Function Charts	Workspace for Sequential Function Charts (SFC). Same icon is displayed next to the name of each SFC.
	Vision Templates (shared)	Workspace for the <i>shared</i> Vision Templates . Same icon is displayed next to the name of each Template.
	Properties	Opens the Project Properties Editor dialog box.
<h3> Scripting</h3>		
	Project Library	Workspace for the Project Library.
	Script	Icon for an individual script. Appears next to the name of the script.
	Gateway Events	Scripts for events that execute on the Gateway.
<h3> Perspective</h3>		
	Perspective Session Events	Scripts for events that occur during a Perspective Session .
	Styles	Styles folder within Perspective.
	Views	Views folder within Perspective.
	View	Icon for an individual View. Appears next to the name of the View.
	Edit View	Edits a View within Perspective.
Additional symbols (Badge icons) can show up to the right of these items. See the Perspective Designer Interface page for an explanation.		
<h3> Transaction Groups</h3>		
	Standard Group	Icon for a block Standard Group.
	Standard Group - Disabled	Indicates that capability to create Standard Groups is disabled.
	Block Group	Icon for a Block Transaction Group.
	Block Group Disabled	Indicates that capability to create Block Groups is disabled.
	Historical Group	Icon for a Historical Group.
	Historical Group -Disabled	Indicates that capability to create Historical Groups is disabled.
	Stored Procedure Group	Icon for a block Stored Procedure Group.
	Stored Procedure Group - -Disabled	Indicates that capability to create Stored Procedure Groups is disabled.

 Vision		
	Client Events	Scripts for events that execute in Vision Clients.
	Windows	Folder for Windows within the Vision Module.
	Window	Icon for a Main Window. Appears next to the Window name.
	Popup Window	Icon for a Popup Window. Appears next to the Popup Window name.
	Docked Window	Icon for a Docked Window. Appears next to the Docked Window name.
	Templates (project)	Workspace for the <i>Project-level</i> Vision Templates. Same icon is displayed next to the name of each Template.
Additional symbols (Badge icons) can show up to the right of these items. See the Vision Designer Interface page for an explanation.		
 Named Queries		
	Named Query	Create new Named Query.
	Named Query - Disabled	Indicates that access to Global resources is disabled.
	Named Query - Editing	Icon for an individual Named Query. Clicking opens the Named Query for editing.
 Reports		
	Report	Create new Report.
	Report - Disabled	Indicates that capability to create a Report is disabled.
	Report - Editing	Icon for an individual report. Clicking opens the report for editing.
Right-Click Menu		
	Rename	Renames the current selection.
	Cut	Cuts the current selection into the clipboard.
	Copy	Copies the current selection into the clipboard.
	Copy Path	Copies the current path into the clipboard.
	Paste	Pastes the content in the clipboard into the selected context.
	Delete	Deletes the current selection.
	Protect	Locks the individual project resource from inside Designer.

Designer Symbol Reference

This reference section provides tables showing the symbols used in Ignition Designer along with the symbol name and description.

General Designer Navigation

Symbol	Name	Description
	About	Displays the About Ignition Designer screen, which includes a list of currently installed modules and their versions.
	Add	General add button used in many areas of the Designer, such as when adding a new style, new page configuration, or new property.
	Add Child	
	Add Custom Property	Adds a custom property to a Perspective component or to a Session.
	Add Sibling	
	Arrows: Down, Left, Right, Up	Navigation arrows.
	Bitwise And	
	Bitwise Lshift	
	Bitwise Not	
	Bitwise Or	
	Bitwise Rshift	
	Bitwise Xor	
	Border Style	
	Breakpoint	
	Bug	
	Calendar	
	Carat right	
	Check	
	Collapse	
	Collapse All	
	Color Chooser	

	Column	
	Column Add	
	Column Delete	
	Comments	
	Comms Off	Gateway communication is disabled, such as queries and Tag subscriptions.
	Comms Read	Read-only communication operations allowed, such as SELECT queries and Tag values.
	Comms Read-Write	Full read/write Gateway communication enabled.
	Console	Opens the Output Console.
	Controls Back	
	Controls Backward	
	Controls Forward	
	Controls Front	
	Controls Pause	
	Controls Stop	
	Coordinate	
	Copy	Copies the current selection to the clipboard.
	Corners Straight	
	Currency	
	Cut	Cuts the current selection into the clipboard.
	Database	
	Database Query Browser	Opens the database query browser window.
	Dataset Browser	
	Debug	
	Delete	Deletes selected item, such as a component or Tag.
	Diagnostics	Displays the Diagnostics screen, which includes Client Performance statistics and error log viewing.
	Disabled	Indicates an option is disabled.
	Duplicate	Duplicates current selection.

		
	Edit	Edit Tag.
	Error	
	Execute	
	Exit	Exit and close Designer.
	Expand	
	Expand All	
	Export Project	Export a backup copy of this Project to a project file (.proj), or export individual resources.
	Export Global	Export a backup copy of the global project, or export individual global resources, such as Alarm Pipelines.
	Find-Replace	Opens the Find/Replace window where you can search in Pipelines, SQL Bridge, Scripting, Tags, Templates, and/or Windows.
	Folder New	Creates a new folder.
	Folder Parent	Go to parent folder.
	Folder Search	Searches for a folder.
	Frame Close	Closes the open frame.
	Frame Dock	
	Frame Pin	
	Frame Unpin	
	Frame Toggle Float	
	Function	
	Function Browser	
	Functions	
	Gateway	
	Gateway Connected	
	Gateway Connection Lost	
	Gateway Connection Unknown	
	Gateway Disabled	
	Group Convert Container	

	Help	Launches the User Manual in a web browser.
	History	
	Horizontal Center	Horizontally center one or more components relative to their parent.
	Image	
	Image Browser	
	Image Management	Opens the Image Management window.
	Import Project	Import resource(s) from a project backup file (.proj).
	Key	
	Key Disabled	
	Launch Applet	Launch this project in applet mode.
	Launch Fullscreen	Launch this Project in full-screen mode.
	Launch Window	Launch the published version of this project in windowed mode.
	Layout Settings	
	Light Bulb	
	Lock	
	Logic And	
	Logic Equal	
	Logic Greater Than	
	Logic Greater Than or Equal To	
	Logic Less Than	
	Logic Less Than or Equal To	
	Logic Not	
	Logic Not Equal	
	Logic Or	
	Merge Changes	
	Monitor	
	Monitor Off	

	More Horizontal	
	More Vertical	
	Node	
	Operator Addition	
	Operator Division	
	Operator Modules	
	Operator Multiplication	
	Operator Power	
	Operators	
	Operator Subtraction	
	Overlay	
	Overlay Disable	
	Palette	
	Panels	
	Panels Reset	Resets the Designer's panel and toolbar layout to the default.
	Parameter In	
	Parameter In Out	
	Parameter Out	
	Parameters	
	Path Connect Curved	
	Path Connect Straight	
	Path Curved	
	Path Disconnect	
	Path Fill	
	Path Intersect	
	Path Perpendicular	
	Path Remove Point	
	Path Straight	

	Percent	
	Phone	
	Preview	Toggles between Preview mode and Design mode.
	Project	
	Project New	Launches a new Project in the Designer.
	Project Update	Merges any new changes on the Gateway into the current Project.
	Property Details	
	Property Transient	
	QR Code	
	Quality Error	
	Quality Pending	
	Quality Unknown	
	Question Answer	Opens the Tip of the Day window.
	Redo	Redo the last action.
	Refresh	
	Report	
	Revise	
	Resize lr	
	Resize tl br	
	Resize tr bl	
	Row	
	Row Add	
	Row Delete	
	Save	
	Save As	Saves the current project as a new project.
	Save Publish	Saves the Project's changes in the Gateway and publishes the Project.

	Script Configure	Configure scripts for the selected component, such as event handlers.
	Script Console	Opens the Script Console.
	Scrollbar Down	
	Scrollbar Left	
	Scrollbar Right	
	Scrollbar Up	
	Scrollbar Lock	
	Search	
	Search dd	
	Security Settings	Shows the security settings for the selected component.
	Select All	Selects all components that are siblings of the selected component.
	Select Type	Selects all components that are siblings to the selected component and the same component type.
	Select Same Type in Window (Vision only)	Selects all components in a window that are the same kind of component as the selected one.
	Settings	
	Size Position	Type exact size and position parameters for the selected component or window.
	Sort All	
	Sort Ascending	Sorts selection in ascending order.
	Sort A to Z	Sorts selection alphabetically (a to z).
	Sort Basic	
	Sort Categorized	
	Sort Descending	Sorts selection in descending order.
	Sort Expert	
	Sort Standard	
	Sort Z to A	Sorts selection from z to a (reverse alphabetical).
	Start Open	
	Styles Config	
	Symbol Factory	Launches the Symbol Factory .

	Table	
	Testing	
	Text Area	
	Text Bold	
	Text Center	
	Text Formatting	
	Text Italic	
	Text Justified	
	Text Left	
	Text Right	
	Text Subscript	
	Text Superscript	
	Text Underline	
	Translations	Opens the Translation manager.
	Tree Collapsed	
	Tree Expanded	
	Tree Node	
	Undo	Undo the last action.
	Ungroup	Break apart the selected component group into its child components.
	Unlock	
	Upload	
	Vertical Center	Vertically center one or more components relative to their parent.
	Warning	
	Welcome	Displays the Designer Welcome screen.
	Window Docked	Opens a new docked Window.
	Window Main	Opens a new main Window
	Window Popup	Opens a new popup Window.

	Zoom In	Zoom into the currently open Window.
	Zoom Out	Zoom out of the currently open Window.
	Zoom Reset	Reset zoom level to 100%.

Perspective Designer Navigation

Symbol	Name	Description
	Add	General add button used in many areas of the Designer, such as when adding a new style, new page configuration, or new property.
	Add Child	
	Add Custom Property	Adds a custom property to a Perspective component or to a Session.
	Add Sibling	
	Arrows: Down, Left, Right, Up	Navigation arrows.
	Binding	Click to set a binding to a Tag.
	Bitwise And	
	Bitwise Lshift	
	Bitwise Not	
	Bitwise Or	
	Bitwise Rshift	
	Bitwise Xor	
	Border Style	
	Breakpoint	
	Bug	
	Calendar	
	Carat right	
	Check	
	Clear Erase	

	Collapse	
	Collapse All	
	Column	
	Column Add	
	Column Delete	
	Comments	
	Controls Back	
	Controls Backward	
	Controls Forward	
	Controls Front	
	Controls Pause	
	Controls Stop	
	Coordinate	
	Copy	Copies the current selection to the clipboard.
	Corners Straight	
	Currency	
	Customizer	Shows the default component customizer for the selected component.
	Cut	Cuts the current selection into the clipboard.
	Database	
	Database Query Browser	Opens the database query browser window.
	Dataset Browser	
	Debug	
	Delete	Deletes selected item, such as a component or Tag.
	Diagnostics	Displays the Diagnostics screen, which includes Client Performance statistics and error log viewing.
	Disabled	Indicates an option is disabled.
	Duplicate	Duplicates current selection.

	Edit	Edit Tag.
	Error	
	Execute	
	Exit	Exit and close Designer.
	Expand	
	Expand All	
	Export Project	Export a backup copy of this Project to a project file (.proj), or export individual resources.
	Export Global	Export a backup copy of the global project, or export individual global resources, such as Alarm Pipelines.
	Folder New	Creates a new folder.
	Folder Parent	Go to parent folder.
	Folder Search	Searches for a folder.
	Frame Close	Closes the open frame.
	Frame Dock	
	Frame Pin	
	Frame Unpin	
	Frame Toggle Float	
	Function	
	Function Browser	
	Functions	
	Gateway	
	Gateway Connected	
	Gateway Connection Lost	
	Gateway Connection Unknown	
	Gateway Disabled	
	Group	Combine the selected components in a group.
	Group Convert Container	
	History	

	Horizontal Center	Horizontally center one or more components relative to their parent.
	Image	
	Image Browser	
	Image Management	Opens the Image Management window.
	Key	
	Key Disabled	
	Launch Applet	Launch this project in applet mode.
	Launch Fullscreen	Launch this Project in full-screen mode.
	Launch Window	Launch the published version of this project in windowed mode.
	Layout Settings	Configure layout constraints for the selected component(s)
	Light Bulb	
	Lock	
	Logic And	
	Logic Equal	
	Logic Greater Than	
	Logic Greater Than or Equal To	
	Logic Less Than	
	Logic Less Than or Equal To	
	Logic Not	
	Logic Not Equal	
	Logic Or	
	Merge Changes	p
	Monitor	
	Monitor Off	
	More Horizontal	
	More Vertical	
	Node	

	Operator Addition	
	Operator Division	
	Operator Modules	
	Operator Multiplication	
	Operator Power	
	Operators	
	Operator Subtraction	
	Overlay	
	Overlay Disable	
	Palette	
	Parameter In	
	Parameter In Out	
	Parameter Out	
	Parameters	
	Paste	Pastes the contents of the clipboard into the current context.
	Paste Cancel	
	Paste Immediate	
	Percent	
	Phone	
	Preview	Toggles between Preview mode and Design mode.
	Project	
	Project New	Launches a new Project in the Designer.
	Project Update	Merges any new changes on the Gateway into the current Project.
	Property Details	
	Property Transient	
	QR Code	
	Quality Error	

	Quality Pending	
	Quality Unknown	
	Question Answer	Opens the Tip of the Day window.
	Redo	Redo the last action.
	Refresh	
	Report	
	Revise	
	Resize lr	
	Resize tl br	
	Resize tr bl	
	Rollback	Rolls back to a previously saved version of the project.
	Rotate Bottom Left	
	Rotate Bottom Right	
	Rotate Left	Rotate the selected components 90 degrees left.
	Rotate Right	Rotate the selected components 90 degrees right.
	Rotate Top Left	
	Rotate Top Right	
	Row	
	Row Add	
	Row Delete	
	Send Back	Moves the selected components to the back of the z-order.
	Send Backward	Moves the selected components backward in the z-order.
	Send Forward	Moves the selected components forward in the z-order.
	Send Front	Moves the selected components to the front of the z-order.
	Size Position	Type exact size and position parameters for the selected component or window.

	Tag Groups (formerly Scan Classes)	Edit Tag Groups.
	Script	
	Script Configure	Configure scripts for the selected component, such as event handlers.
	Scrollbar Down	
	Scrollbar Left	
	Scrollbar Right	
	Scrollbar Up	
	Scrollbar Lock	
	Search	
	Search dd	
	Send Back (Perspective only)	Moves the selected components to the back of the z-order.
	Send Backward (Perspective only)	Moves the selected components backward in the z-order.
	Send Forward (Perspective only)	Moves the selected components forward in the z-order.
	Send Front (Perspective only)	Moves the selected components to the front of the z-order.
	Settings	
	Size Position	Type exact size and position parameters for the selected component or window.
	Sort All	
	Sort Ascending	Sorts selection in ascending order.
	Sort A to Z	Sorts selection alphabetically (a to z).
	Sort Basic	
	Sort Categorized	
	Sort Descending	Sorts selection in descending order.
	Sort Expert	
	Sort Standard	
	Sort Z to A	Sorts selection from z to a (reverse alphabetical).
	Start Open	
	Styles Config	

	Table	
	Testing	
	Text Area	
	Text Bold	
	Text Center	
	Text Formatting	
	Text Italic	
	Text Justified	
	Text Left	
	Text Right	
	Text Subscript	
	Text Superscript	
	Text Underline	
	Tree Collapsed	
	Tree Expanded	
	Tree Folder	Folder, closed.
	Tree Folder Open	Folder, open.
	Undo	Undo the last action.
	Unlock	
	Upload	
	Vertical Center	Vertically center one or more components relative to their parent.
	Warning	
	Window Docked	Opens a new docked Window.
	Window Main	Opens a new main Window
	Window Popup	Opens a new popup Window.
	Zoom In	Zoom into the currently open Window.
	Zoom Out	Zoom out of the currently open Window.



Zoom Reset

Reset zoom level to 100%.

Vision Designer Navigation

Symbol	Name	Description
	Align Bottom (g)	Align two or more components along their bottom edge.
	Align Top (g)	Align two or more components along their top edge.
	Align Column (g)	Align two or more components into a vertical stack (column).
	Align Horizontal (g) (g)	Align two or more components horizontally along their centerpoints.
	Align Left (g)	Align two or more components along their left edge.
	Align Right (g)	Align two or more components along their right edge.
	Align Row (g)	Align two or more components along into a horizontal row.
	Align Vertical (g)	Align two or more components vertically along their centerpoints.
	Binding	Click to set a binding to a Tag.
	Clear Erase	Clear all pixels.
	Color Chooser	
	Customizer	Shows the default component customizer for the selected component.
	Drawing Tool: Arrow	Create single or double-sided arrow shapes
	Drawing Tool: Circle	Creates and edits circles and ellipses.
	Drawing Tool: Eyedropper	Used to set the selected shape(s) and/or component(s) foreground/background or stroke/fill colors by pulling the colors from somewhere else in the window.
	Drawing Tool: Gradient	When a shape is created with the Fill paint property set as a gradient, you can use this Gradient tool to adjust the angle and direction of the gradient.
	Drawing Tool: Line	Draws lines, arbitrary polygons, or curved paths.
	Drawing Tool: Path	Directly edit a path's vertices and curve handles Nodes can be moved, deleted, or added to modify the shape.
	Drawing Tool: Pencil	Draws freehand lines and shapes.
	Drawing Tool: Polygon	Creates and edits polygons and stars.
	Drawing Tool: Rectangle	Creates and edits rectangular and square shapes.
	Drawing Tool: Selection	When this tool is active, you can select shapes and components.

	Group	Combine the selected components in a group.
	Horizontal Center	Horizontally center one or more components relative to their parent.
	Mirror Horizontal	Mirror selected objects horizontally.
	Mirror Vertical	Mirror selected objects vertically.
	Palette	Color palette for drawing tools.
	Paste	Pastes the contents of the clipboard into the current context.
	Paste Cancel	Cancels a pending paste operation
	Paste Immediate	Pastes the contents of the clipboard onto a window immediately, at whatever location they were cut/copied at.
	Path Connect Curved	
	Path Connect Straight	
	Path Curved	
	Path Disconnect	
	Path Fill	
	Path Intersect	
	Path Perpendicular	
	Path Remove Point	
	Path Straight	
	Rotate Left	Rotate the selected components 90 degrees left.
	Rotate Right	Rotate the selected components 90 degrees right.
	Select All	Selects all components that are siblings of the selected component.
	Select Type	Selects all components that are siblings to the selected component and the same component type.
	Select Same Type in Window	Selects all components in a window that are the same kind of component as the selected one.
	Shape Difference	Alters the first selected shape so that the other selected shapes will be subtracted from it. V
	Rotate Bottom Left	
	Rotate Bottom Right	
	Rotate Top Left	

	Rotate Top Right	
	Shape Division	Cuts the first selected shape into multiple shapes, divided on the edges of the other shape(s). V
	Shape Exclusion	Alters the first selected shape to become the area that is contained in exactly one shape. V
	Shape Intersection	Creates an intersection of the selected shapes. The first selected shape will become the the intersection of the selected shapes (the area that all shapes share).
	Shape Path	Convert shape to a simple path.
	Shape Stroke	Convert the selected shape into a new shape defined by its stroke.
	Shape Union	Merges the selected shapes. The first selected shape will become the the union of all selected shapes. V
	Size Position	Type exact size and position parameters for the selected component or window.
	Touchscreen	Toggles touchscreen mode for the Designer for testing purposes.
	Ungroup	Break apart the selected component group into its child components.

Ignition Gateway File Reference.

When you install Ignition, it will create a directory that will contain many files. It can be important to understand some of these files and what they are used for. This is not an exhaustive list of files.

- Ignition - Directory which contains all of the files in the Gateway. The default install directory for windows is C:\Program Files\Inductive Automation
 - data
 - db
 - projects
 - gateway.xml
 - lib
 - logs
 - user-lib
 - jdbc
 - modules
 - pylib
 - install-ignition.bat
 - start-ignition.bat
 - stop-ignition.bat
 - uninstall-ignition.bat

Data Type Formatting Reference

This is a reference page for Date, Number, and String formatting.

Date Format

Date as a string, formatted according to a pattern. The pattern is a format that is full of various placeholders that will display different parts of the date. These are case-sensitive! These placeholders can be repeated for a different effect. For example, M will give you 1-12, MM will give you 01-12, MMM will give you Jan-Dec, MMMM will give you January-December.

Functions that use this date formatting include:

- [system.date.format](#)
- [system.dataset.formatDates](#)

Symbol	Description	Presentation	Examples	Other Notes
a	Am/Pm marker	Text	PM	
D	Day in year	Number	189	
d	Day in month	Number	10	
E	Day name in week	Text	EEEE=Tuesday; E=Tue	
F	Day of week in month	Number	2	2nd Sunday of the month
G	Era designator	Text	G=AD	
H	Hour in day (0-23)	Number	0	
h	Hour in am/pm (1-12)	Number	12	
k	Hour in day (1-24)	Number	24	
K	Hour in am/pm (0-11)	Number	0	
M	Month in year	Month	MMMM=July; MMM=Jul; MM=07	
m	Minute in hour	Number	30	
s	Second in minute	Number	55	
S	Millisecond	Number	978	
u	Day number of week	Number	1	(1 = Monday, ..., 7 = Sunday)
w	Week in year	Number	27	If Dec31 is mid-week, it will be in week 1 of the next year
W	Week in month	Number	2	
X	Time zone	ISO 8601 time zone	X=-08; XX=-0800; XXX=-08:00	
y	Year	Year	yyyy=1996; yy=96	Lowercase y is the most commonly used year symbol
Y	Week year	Year	YYYY=2009; YY=09	Capital Y gives the year based on weeks (ie. changes to the new year up to a week early)
z	Time zone	General time zone	zzzz=Pacific Standard Time; Z=PST	
Z	Time zone	RFC 822 time zone	Z=-0800	

Number Format

Returns a string version of the number argument, formatted as specified by the pattern string. This is commonly used to specify the number of decimal places to display, but can be used for more advanced formatting as well. The pattern string is a numeric format string, which may include any of these characters that instruct it how to format the number.

Functions that use this number formatting include:

- [numberFormat](#)

Symbol	Description
0	Specifies a required digit.
#	Specifies an optional digit.
,	The grouping separator.
.	The decimal separator.
-	A minus sign.
E	Scientific notation.
;	Used to separate positive and negative patterns. The negative subpattern will only be used to specify the prefix and suffix. The number of digits, , minimal digits, and other characteristics are all the same as the positive pattern.
%	Multiplies the value by 100 and shows as a percent.
'	Used to quote special characters.

Example

This table shows some numbers, and the result of using various format strings to format them.

Number	Pattern	Result
5	0	5
5	0.0	5.0
5	00.0	05.0
123	#,##0	123
1024	#,##0	1,024
1337	#,##0.#	1,337
1337.57	#,##0.#	1,337.6
87.32	#,##0.0000	87.3200
-1234	#,##0	-1,234
-1234	#,##0;(#)	(1,234)
4096	0.###E0	4.096E3
.348	#.00%	34.80%
34.8	#0.00%'	34.80%

String Format

Functions that use this data formatting include:

- [stringFormat](#)

Date/Time Formatting Elements

	Description
--	-------------

Date/Time Formatting Element Suffix	
'H'	Hour of the day for the 24-hour clock, formatted as two digits with a leading zero where necessary, i.e., 00 - 23.
'I'	Hour of the 12-hour clock, formatted as two digits with a leading zero as necessary, i.e., 01 - 12.
'k'	Hour of the day for the 24-hour clock, i.e., 0-24.
'K'	Hour of the 12-hour clock, i.e., 1 - 12.
'M'	Minute within the hour formatted as two digits with a leading zero where necessary, i.e., 00 - 59.
'S'	Seconds within a minute, formatted as two digits with a leading zero where necessary, i.e., 00 - 59.
'L'	Millisecond within the second formatted as three digits with leading zeros as necessary, i.e., 000-999.
'B'	Locale-specific full month name i.e. "January", "March".
'b'	Locale-specific abbreviated month name i.e. "Jan", "Mar".
'A'	Full name of the day of the week i.e. "Monday".
'a'	Abbreviated name of the day of the week i.e. "Mon".
'Y'	Year formatted as a 4 digit numeric value with leading zeros where necessary i.e. 0005 would be the year 5 in the Gregorian calendar.
'y'	Last two digits of the year formatted with leading zeros where necessary.
'j'	Day of the year formatted as three digits with leading zeros where necessary, i.e., 001 - 366 for the Gregorian calendar.
'm'	Month, formatted as a two digit number with leading zeros where necessary, i.e., 01-13.
'd'	Day of the month formatted as two digit number with leading zeros where necessary, i.e., 00 - 31.
'e'	Day of the month formatted as two digits, i.e., 0-31.

Formatting Elements

Element Character	Data Type to Substitute Element Character	Description
'b', 'B'	Boolean	If the corresponding argument <i>arg</i> is NULL, then a False is substituted into the format string. If <i>arg</i> is boolean, then the string conversion of <i>arg</i> will be substituted into the format string. For every other condition, True is substituted into the format string.
's', 'S'	String	The string value <i>arg</i> is substituted into the formatted sting.
'c', 'C'	Character	The unicode value <i>arg</i> is substituted into the formatted string.
'd'	Integral	The decimal value <i>arg</i> is substituted into the formatted string.
'f'	Floating Point	The floating point value <i>arg</i> is substituted into the formatted string.
't', 'T'	Date/Time	This is the prefix for date/time arg values to be used for string formatting.

Scripting Object Reference

The following sections describe the structure of some prominent objects returned or expected by Ignition system functions. Not all classes, fields, and methods are included in the lists below. For a fully comprehensive reference of all Ignition types, refer to the Javadocs in the [Ignition SDK Programmers Guide](#).

Alarming

AlarmEvent

Used By: Alarm Pipelines, Tag Event Scripts, Vision Extension Functions

See [Alarm Event Properties Reference](#).

AlarmQueryResult

Used By: `system.alarm.queryStatus`, `system.alarm.queryJournal`

The `AlarmQueryResult` object represents a collection of `AlarmEvent` objects with some additional helper methods.

Methods		
Name	Returns	Description
<code>getDataset()</code>	<code>dataset</code>	Returns the alarms as a dataset. Columns are: <ul style="list-style-type: none">• Event Id (UUID of the alarm)• Source Path• Display Path• Event Time• State (integer)• Priority (integer)
<code>getAssociatedData(String uuid)</code>	<code>dataset</code>	Returns the associated data of an event as a dataset.
<code>AlarmEvent(String uuid)</code>	<code>alarmEvent</code>	Returns the <code>alarmEvent</code> object for the specified event UUID.

ShelvedPath

Used By: `system.alarm.getShelvedPaths`

This class provides information about an alarm that has been "shelved", including the alarm path, when it was shelved, and by whom.

Methods		
Name	Returns	Description
<code>getPath()</code>	<code>path</code>	Returns a <code>Path</code> object representing the path to the alarm. Can be cast to a string.
<code>getExpiration()</code>	<code>date</code>	Returns the time that the shelf expires.
<code>getUser()</code>	<code>QualifiedPath</code>	Returns the actual <code>AlarmEvent</code> object for the specified event UUID.
<code>isExpired()</code>	<code>boolean</code>	Returns True if the shelf has expired for the alarm; otherwise returns False.

Datasets and PyDatasets

Used By: Many facets of Ignition including queries, Vision and Perspective components, and Tags

On this page ...

- Alarming
 - AlarmEvent
 - AlarmQueryResult
 - ShelvedPath
- Datasets and PyDatasets
- Queries
 - SProcCall
- Tags
 - Results
- Perspective
- Misc Objects
 - Qualified Value
 - QualityCode

See [Datasets](#).

Queries

SProcCall

Used By: [system.db.createSProcCall](#), [system.db.execSProcCall](#)

See [system.db.createSProcCall](#).

Tags

Results

Used By: [system.tag.browse](#), [system.tag.browseHistoricalTags](#)

Methods		
Name	Ret ur ns	Description
getResults()	list	Returns a list of node dictionaries, one for each Tag (including folders and UDT instances) in the result set. See Tag Dictionary Keys below for a summary of available keys in the Tag dictionaries.
getReturnedSize()	integer	Returns the number of results contained in this object. If different than getTotalAvailableSize(), only a partial browse was performed, and the continuation point should be non-null.

Tag Dictionary Keys

All Nodes (Tags, Folders, UDT Definitions, and UDT Instances)		
Name	Type	Description
fullPath	BasicTagPath	A fully qualified Tag path to the node, including the name of the node. The value returned by this key is a BasicTagPath. However, Python's str() method can be used to convert the path to a string.
hasChildren	boolean	Represents whether the node contains sub-nodes, such as folders and UDT definitions.
name	string	The name of the node.
tagType	string	The type of the node. Possible values are: <ul style="list-style-type: none">• AtomicTag: Represents a standard Tag.• Folder: Represents a Tag folder.• UdtType: Represents a UDT definition.• UdtInstance: Represents a UDT instance.
Tags (tagType = AtomicTag)		
Name	Type	Description
dataType	string	The data type of the Tag.
valueSource	string	Represents how the Tag derives its value.
value	Varies	The last known qualified value on the Tag.
UDT Definitions and Instances (tagType = UdtType or UdtInstance)		
Name	Type	Description
typId	string	Represents the parent data type for the UDT.

Perspective

See [Perspective Component Methods](#).

Misc Objects

Qualified Value

Used By: Many facets of Ignition including Tag Event Scripts, various [system.tag](#) functions, and Perspective Property Change scripts

The **QualifiedValue** class is a representation of an Ignition value often used to represent a Tag's current condition (as in the result of a Tag read) and is meant to provide more context than a value alone. The properties and methods on a **QualifiedValue** object are as follows:

Properties		
Name	Type	Description
value	varies	The data associated with the QualifiedValue object. Type depends on the type of the underlying value.
quality	qualityCode	The quality associated with the returned value. See QualityCode .
timestamp	date	The time associated with the returned value.
Methods		
Name	Returns	Description
getValue()	varies	Gets the data associated with the QualifiedValue object. Type depends on the type of the underlying value.
getQuality()	qualityCode	Gets the quality associated with the returned value. See QualityCode .
getTimestamp()	date	Gets the time associated with the returned value.

QualityCode

Used By: QualifiedValue Objects

The **QualityCode** class describes a quality for a value.

Methods		
Name	Returns	Description
getCode()	integer	Returns the 32-bit integer code associated with the quality. For more information on quality codes, including a list of available codes, see Tag Quality and Overlays .
getDiagnosticMessage()	string	Returns the diagnostic message associated with this quality, if present. Otherwise, returns None.
isGood()	boolean	Returns True if this code represents a Good quality level. Otherwise, returns False.
isUncertain()	boolean	Returns True if this code represents an Uncertain quality level. Otherwise, returns False. Note that Bad and Error quality levels return False.
isBad()	boolean	Returns True if this code represents a Bad quality level. Otherwise, returns False. Note that Uncertain and Error quality levels return False.
isError()	boolean	Returns True if this code represents an Error quality level. Otherwise, returns False. Note that Bad and Uncertain quality levels return False.
isNotGood()	boolean	Returns True if this code represents an Uncertain, Bad, or Error quality level. Otherwise, returns False.
isBadOrError()	boolean	Returns True if this code represents a Bad or Error quality level. Otherwise, returns False.
toValue()	Qualified Value	Returns this quality as the quality component of a QualifiedValue object.
is(QualityCode other)	boolean	Compares the integer quality codes of another QualityCode object to this one. Returns True if they match.

isNot(QualityCode)	boolean other)	Compares the integer quality codes of another QualityCode object to this one. Returns True if they do not match.
--------------------------------------	-------------------	--

Alarm Event Properties Reference

Alarm Event properties provide a lot of information regarding an alarm event and fall into several categories: General Alarm properties, Event Alarm properties, and Runtime Event properties. General Alarm properties are automatically configured with default settings when the module is first installed. Event Alarm properties are properties that occur when an event happens. Runtime Properties are properties that are only present while the alarm event is in memory.

Alarm Event properties are described below. If you are looking for information on the properties associated with configuring an alarm, please see the [Tag Properties](#) page.

Gateway General Alarm Properties

General alarm properties are configured during initial setup.

1. Go to the Config section of the Gateway Webpage, and choose **Alarming > General** from the menu on the left. The General Alarms Settings page will appear.
2. Set the general alarm properties shown in the following table:

Alarm Evaluation	
Live Event Limit	Default is 5. The number of "live" events (active or unacknowledged) that can exist for a single alarm at a given time. When surpassed, older events will be acknowledged automatically by the system. This means as an alarm cycles on and off, Ignition will keep track of the last five times the alarm event happened until the user acknowledges them. This does not store history for those events.
Event Suppression	
Continuous Event Detection Window (min)	Default is 10. The amount of time to store events before shutdown to prevent new duplicate events from being created on startup. This setting prevents unacknowledged active events from being generated due to reboot. If set to 0, will not be used.
Notify Initial Events	Default is false. If false, active alarms caused by the "initial state" (that is, the first value checked after being created, or after the enabled state changes) won't be sent to the notification system. This means if you add an alarm to a Tag, a notification won't be immediately sent when the new state is created.

Alarm Event Properties

An alarm event is an instance of an alarm that was witnessed by the alarming system. For example, when the value on a Tag meets the criteria for an alarm on the same Tag, an alarm event is created. That same event is then used to compile information about the event, such as when it transitions from an active state to a cleared state.

When writing scripts that reference properties on an alarm event, the Binding/Scripting Key is case insensitive, meaning capitalization does not matter.

Note: The value of properties on Alarm Events represent the value at the time the event was created. Thus, changing the value on an alarm property after one or more events were created will not retroactively change the property value on prior events.

Alarm Event Configuration Properties

The table below represents properties on an alarm event that are derived by the configuration of the alarm.

Event Properties	Binding/Scripting Key	Description	Data Type				
Name	name	The name of the Alarm.	String				
Enabled	enabled	Specifies whether or not this alarm is evaluated by the system. Set to False to turn off the alarm state and all associated actions.	Boolean				
Priority	priority	<p>The priority (or severity) of the alarm. Used for sorting/filtering. Numerical values are associated with each priority to make comparison easier. This property can also be referenced as a string with the following priority names.</p> <table border="1"><thead><tr><th>Value</th><th>Priority</th></tr></thead><tbody><tr><td>0</td><td>Diagnostic</td></tr></tbody></table>	Value	Priority	0	Diagnostic	Integer
Value	Priority						
0	Diagnostic						

On this page ...

- [Gateway General Alarm Properties](#)
- [Alarm Event Properties](#)
 - [Alarm Event Configuration Properties](#)
 - [Alarm Runtime Properties](#)
- [The AlarmEvent Object](#)

		<table border="1"> <tr><td>1</td><td>Low</td></tr> <tr><td>2</td><td>Medium</td></tr> <tr><td>3</td><td>High</td></tr> <tr><td>4</td><td>Critical</td></tr> </table>	1	Low	2	Medium	3	High	4	Critical	
1	Low										
2	Medium										
3	High										
4	Critical										
Display Path	displayPath	An optional path (separated by "/" characters) that is used for display and browsing purposes. Defaults to an empty string.	String								
Active Pipeline	activePipeline	The pipeline (if any) that will be used to process active events generated by the alarm.	String								
Clear Pipeline	clearPipeline	The pipeline (if any) that will be used to clear events generated by the alarm. Used when the alarm goes into the Clear State.	String								
Active Delay	timeOnDelaySeconds	The amount of consecutive seconds that the alarm state must be True before the Tag enters this alarm state.	Double								
Clear Delay	timeOffDelaySeconds	The amount of consecutive seconds that the alarm state must be False before the Tag exits this alarm state.	Double								
Notes	notes	A string that can act as documentation about the alarm.	String								
Ack Notes	ackNotes	Notes added by user that acknowledged the alarm when it went active.	String								
Associated Data*	The key is a string literal that matched the name of the Associated Data.	Alarm Associated Data can also represented on alarm event.	String								

Alarm Runtime Properties

The table below represents properties that represent the that are provided to the event

Runtime Property	Binding /Scripting Key	Description	Data Type										
Is Initial Event?	IsInitialEvent	Set to "true" when the event is caused by the initial state of the alarm.	Boolean										
System Acknowledgment	SystemAck	Set to "true" when the alarm has been acknowledged by the system, due to an overflow of the "live event queue". Live events are alarm events that are active or not acknowledged, and are limited for each alarm by the general alarm settings.	Boolean										
Shelf Expiration	ShelfExpiration	When the shelf will expire for this event.	Integer										
Is Shelved	IsShelved	A boolean value that reports if the alarm is currently shelved.	Boolean										
Event Canceled	EventCancelled	If set, the event will drop out as soon as possible from the pipelines.	Boolean										
Event Id	EventId	The Unique ID (UUID) of this alarm event. Each event gets a completely unique id.	String										
Source	Source	The qualified path to the item that generated this event. Includes the Tag Provider, Tag Path, and the name of the alarm. Example: prov:tagProviderName:/tag:folder/tagName:/alm:alarmName	String										
Display Path Or Source	DisplayPathOrSource	Gets the display path if defined, otherwise, returns the source.	String										
State	State	<p>The current overall state of the alarm.</p> <table border="1"> <thead> <tr> <th>Value</th><th>State</th></tr> </thead> <tbody> <tr><td>0</td><td>Clear and Unacked</td></tr> <tr><td>1</td><td>Clear and Acked</td></tr> <tr><td>2</td><td>Active and Unacked</td></tr> <tr><td>3</td><td>Active and Acked</td></tr> </tbody> </table>	Value	State	0	Clear and Unacked	1	Clear and Acked	2	Active and Unacked	3	Active and Acked	Integer
Value	State												
0	Clear and Unacked												
1	Clear and Acked												
2	Active and Unacked												
3	Active and Acked												
EventState	EventState	The transitional state that caused the current event.	Integer										
		<table border="1"> <thead> <tr> <th>Value</th><th>State</th></tr> </thead> <tbody> <tr><td>0</td><td>Active</td></tr> <tr><td>1</td><td>Clear</td></tr> </tbody> </table>	Value	State	0	Active	1	Clear					
Value	State												
0	Active												
1	Clear												

		2	Acknowledged	
Event Value	EventValue	The value associated with the current event.		Integer
Acknowledgment User	AckUser	The user who acknowledged this event.		String
Is Acknowledged?	IsAcked	"True" if the event has been acknowledged.		Boolean
Is Active?	IsActive	"True" if the event is still active.		Boolean
Is Clear?	IsClear	"True" if the event is not active.		Boolean
Active Time	ActiveTime	A datetime representing when the event became Active.		datetime
Clear Time	ClearTime	A datetime representing when the event became Cleared.		
Acknowledgment Time	AckTime	A datetime representing when the event became Acknowledged.		
Pipeline Transition Count	PipelineTransitionCount	How many transitions the event has made inside of the pipelines.		Integer

The AlarmEvent Object

Some system functions, such as [system.alarm.queryStatus](#), return an AlarmEvent object, which contains many methods to retrieve additional information about the individual alarm. This section details the various methods on an AlarmEvent object.

Many of the functions below return some complex object, as opposed to just a standard Python data type. In these cases, normal Python type-casting can be used to turn the object into a native Python type. For example, `getDisplayPath()` returns a StringPath object, which can easily be converted to a Python string with the `str()` function.

Function	Description	Returned Object
<code>getDisplayPath</code>	Returns the Display Path of the alarm.	StringPath
<code>getDisplayPathOrSource</code>	If the Display Path for the alarm is an empty string, then returns the Source Path, otherwise, returns the Display Path.	Python Unicode
<code>getId</code>	Returns the UUID of the alarm event.	java.util.UUID
<code>getLabel()</code>	The label of the alarm event.	Python Unicode
<code>getLastEventState</code>	Returns the last state of the alarm. Possible return values are Active , Acknowledged , or Cleared . The last event is always returned, so if you're looking to see if an alarm is has been both Acknowledged and Cleared, use <code>getState()</code> instead.	AlarmStateTransition
<code>getName</code>	Returns the name of the alarm.	Python Unicode
<code>getNotes</code>	Returns the value of the Notes property. Returns a None-type object if a Note has not been configured on the Alarm.	Python Unicode or None
<code>getPriority</code>	Returns an AlarmPriority object representing the the Priority of the alarm. Can easily be converted to a String with <code>str()</code> .	AlarmPriority
<code>getSource</code>	Returns the Source path of the alarm.	QualifiedPath
<code>getState</code>	Returns the current state of the alarm: i.e., Active, Unacknowledged.	AlarmState
<code>isAcked</code>	Returns a boolean flag indicating that the alarm has been Acknowledged.	Python boolean
<code>isCleared</code>	Returns a boolean flag indicating that the alarm has been Cleared.	Python boolean
<code>isShelved</code>	Returns a boolean flag indicating that the alarm has been Shelved.	Python boolean

Related Topics ...

- [Alarm Associated Data](#)
- [Alarm Notification Pipelines](#)
- [Tag Properties](#)

Tutorials & Helpful Tricks

This section has pages that include some 'Tips and Tricks'; general information and guides on doing specific tasks within Ignition.

[In This Section ...](#)

Mapping a Network Drive

Windows makes it possible to map drives on network servers to local drive letters so they can be accessed by users as if they were a local drive. The problem, however, is that Windows is not very consistent about how it handles such mapped drives when accessed from a Windows Service, such as the Ignition Gateway. When the service is started manually, the drives will be available, but when the system is rebooted, the service will no longer be able to access them. There may be users that wish to read or write data in Ignition using shared drives, and don't want to manually set up shared drives each time.

Note: Must be using Ignition at least 7.5 and Java Service Wrapper 3.5.4.

To make shared drives available to Ignition on startup, place the following lines in the ignition.conf file, which is located in the data folder of the main Ignition installation folder (usually C:\Program Files\Inductive Automation\Ignition\data for Windows users):

```
//Note that target "Z:" is the drive letter assigned to the mapped drive.  
  
wrapper.share.1.location=\\fileserver\\folder  
wrapper.share.1.target=Z:  
wrapper.share.1.type=DISK
```

Then Stop the Gateway and Start the Gateway. Restarting does not enable this functionality.

Change the appropriate data for location and target to match the computer's actual setup. If your shared drives require authentication, add the following lines, filling in the appropriate data for user, domain, and password:

```
wrapper.ntservice.account=user  
wrapper.ntservice.password=password  
wrapper.share.1.account=domain\\user  
wrapper.share.1.password=password
```

To turn on debugging to see what is causing network share connection issues, make sure to enable wrapper.debug = TRUE by removing the pound (#) sign in front of that line of code. The location of the log file is in the main Ignition installation folder as wrapper.log.

Note: Remember, the Gateway must be stopped then started to get the changes to the ignition.conf file to take effect. Restarting the Gateway will not work.

Other Notes

Un-map drive on shutdown

```
wrapper.share.1.shutdown.unmap=TRUE
```

How often to retry server connection:

```
wrapper.share.1.startup.max_retries=2
```

What interval (seconds) between retries:

```
wrapper.share.1.startup.retry_interval=10
```

Set wrapper to fail to startup if network share not found:

```
wrapper.share.1.startup.failure=SHUTDOWN
```

Troubleshooting:

Here a couple of the common problems that are encountered when mapping network shares:

- **Server not found**

The debug output will show something like this if a drive can't be reached

```
wrapper | Attempting to map the "\\fileserver\folder" share to "S:"...
wrapper | Unable to map "S:". Attempt #1 (The network name cannot be found. (0x43))
wrapper | Attempting to map the "\\fileserver\folder" share to "S:"...
wrapper | Unable to map "S:". Trying to continue. (The network name cannot be found. (0x43))
```

- **Incorrect Login Data**

If the configured account or password are incorrect (or are missing) then the mapping will fail with a message like the following:

```
wrapper | Attempting to map the "\\myfileserver\commonshare" share to "S:"...
wrapper | Unable to map "S:". Trying to continue. (Access is denied. (0x5))
```

Troubleshooting Guides

Troubleshooting Ignition Issues

Here we have compiled a few helpful Troubleshooting Guides to help you diagnose and troubleshoot issues you may have with your Ignition software. They typically include various scenarios that you may find yourself in, and what to do in those situations, as well as some loggers that will be helpful in figuring out what is going wrong.

Loggers

The loggers that are supplied with each guide are used to find and set loggers to a higher logging level, to give more detail to the issue. To use them, you will first want to navigate to the Gateway Status page. Once there, go to the Logs page under Diagnostics.

On this page, you can then click the Settings  button on the right side of the page. This will bring up the Log Configuration window. Here, you can enter in one of the loggers that are listed on the troubleshooting page into the filter, and then change its logging level to either DEBUG or TRACE. This will cause that specific logger to report more detailed logs to the Logs page, allowing you to better diagnose the issue. If you set a parent logger to a specific level, it will set all children of that logger to the same level.

Note: You may find that some of the loggers listed on the Troubleshooting Guide pages are not listed in your gateway. This is normal. The Gateway will add the appropriate loggers as you set systems in your Gateway up.

The screenshot shows the Ignition Log Configuration interface. On the left, there's a sidebar with 'SYSTEMS' and 'CONNECTIONS' sections. The main area is titled 'Log Configuration' and has a 'Loggers' tab selected. It lists several loggers with their levels set to 'INFO': 'AlarmQueryProviderServiceVersionAdapter', 'Alerting', 'Alerting.Notification', 'Alerting.Notification.NotificationManager', and 'Alerting.Storage'. There's a 'Filter' input field and a 'Reset Levels' button.

MDC Keys

MDC Keys were introduced in 7.8.0, and allow you to quickly set all loggers that are related to a specific part of your Gateway. Instead of setting individual loggers to DEBUG, using the MDC key will set a group of loggers to the level you select all at once. Each guide will list out any MDC Keys that will relate to what the guide is covering.

Using MDC Keys is very simple. Again, you will want to navigate to the Gateway Status page, and go to the logs page under Diagnostics. Click on the settings button to bring up the Log Configuration but this time click on the 'Context' tab. Here, there are a few fields where you can enter information. The first field, **Key** is where you will enter in the appropriate MDC Key. The field will list out appropriate Keys for you as you type, making it easier to find what you are looking for. For the **Value** field, you will want to enter in a value that is appropriate for the given MDC Key. This field also will list appropriate values for you to enter. In the example in the image below, with the Key being **database**, the value field has listed out all of the database connection names, making it easy to select which database you want to troubleshoot. Finally, you set the appropriate logging level in the **Set Level** field, typically to **DEBUG** or **TRACE**, and then hit the **Add Level** button. You can add as many of these as you would like, even with the same Key. So in the example below, you can first add my database called **MSSQL**, and then using the database Key again, add in the database called **MySQL**. This allows you to quickly and easily set the appropriate loggers to find out more information of the system you are trying to troubleshoot.

The screenshot shows the 'Log Configuration' dialog with the 'Context' tab selected. It has fields for 'Key' (set to 'database'), 'Value' (with a placeholder 'Enter values'), 'Set Level' (set to 'INFO'), and an 'Add Level' button. Below this is a table with columns 'Key', 'Value', 'Level', and 'Action'. It shows rows for 'Database' with values 'MSSQL' and 'MySQL', and a note 'No Log Properties Configured'. At the bottom, a log viewer shows two entries from a 'FaultedRetryDaemon' process.

Scenarios

Each guide will also include one or more scenarios that you may find yourself in, and some steps you can take to troubleshoot. These can be as simple as determining where the problem actually is, making things run a little smoother when you contact support, to walking you through the problem and how to diagnose and fix it. These can be very useful, but it is important to understand that these are generic troubleshooting guides, and these scenarios might not apply to all systems, nor are they guaranteed to fix all problems.

In This Section ...

Troubleshooting Guide - Alarm Notification

Loggers

This is a list of loggers that will be useful in troubleshooting issues with Alarm Notification.

alarm

alarm.AlarmNotificaitonManager

alarm.JournalManager

alarm.LegacyAdapter

alarm.Execution

alarm.Execution.BoundProperties

alarm.Execution.Deadbands

alarm.Journal

alarm.Journal.Query

alarm.Manager

alarm.Manager.StateManager

alarm.Notification

alarm.Notification.Blocks

alarm.Notification.PipelineManager

alarm.Notification.Sip

alarm.Notification.Pipeline

alarm.Notification.Pipeline.BlockEvaluationContext

alarm.Notification.Voice

alarm.Notification.Voice.Phrases

alarm.Notification.Voice.ScriptDirector

alarm.Notification.Voice.CallManager

alarm.Notification.Voice.CallManager.Agent

alarm.Notification.Voice.CallManager.PhoneCall

GatewayNetwork.Remote.Services.AlarmNotification

MDC Keys

This list of MDC Keys will turn on specific loggers that relate to the Keys context.

- alarm-name
- alarm-notification-profile
- alarm-pipeline
- alarm-source
- roster

On this page ...

- Loggers
 - alarm
 - alarm.AlarmNotificaitonManager
 - alarm.JournalManager
 - alarm.LegacyAdapter
 - alarm.Execution
 - alarm.Execution.BoundProperties
 - alarm.Execution.Deadbands
 - alarm.Journal
 - alarm.Journal.Query
 - alarm.Manager
 - alarm.Manager.StateManager
 - alarm.Notification
 - alarm.Notification.Blocks
 - alarm.Notification.PipelineManager
 - alarm.Notification.Sip
 - alarm.Notification.Pipeline
 - alarm.Notification.Pipeline.BlockEvaluationContext
 - alarm.Notification.Voice
 - alarm.Notification.Voice.Phrases
 - alarm.Notification.Voice.ScriptDirector
 - alarm.Notification.Voice.CallManager
 - alarm.Notification.Voice.CallManager.Agent
 - alarm.Notification.Voice.CallManager.PhoneCall
 - GatewayNetwork.Remote.Services.AlarmNotification
- MDC Keys
- Scenarios

Scenarios

Troubleshooting Guide - Databases

Scenarios

I want to make sure that all of my database connections are valid.

The status of all of your connections can be viewed at a glance; the Overview page has a quick reference in the Connections section, while the Databases page gives a more comprehensive overview.

1. In **Gateway / Status**, navigate to the **Overview** page under the **Systems** category.
2. Under the Connections section, the Databases tab will show the number of connected and faulted databases.

The screenshot shows the Ignition 157 web interface. The top navigation bar includes links for HOME, STATUS (which is highlighted), and CONFIGURE. The left sidebar has sections for SYSTEMS (with Overview selected), CONNECTIONS (with Databases selected), and DIAGNOSTICS. The main content area is titled 'Systems Overview' and contains several cards:

- Architecture**: Displays 'Gateway | Ignition157' with version 7.9.2, license status, and uptime of 7 minutes. It also shows CPU usage at 2% and memory at 151 mb.
- No Redundancy**: A callout suggesting adding a redundant backup gateway.
- No Gateway Network**: A callout suggesting combining multiple gateways into an enterprise network.
- Connections**: A section with three cards:
 - Designer Sessions**: 0 open.
 - Databases**: 1 / 3 connected. This card is highlighted with a red border and a red arrow points to it from the left sidebar's CONNECTIONS section. The status message includes a warning: "⚠ 2 databases faulted".
 - OPC Connections**: 1 / 1 connected.
- Enviror**: A sidebar with various system status indicators.
- System**: A sidebar with system management links.

3. You may also click on that tab which will take you to the Databases page, where you can see a list of all of your database connections and any issues each one might be having.

The screenshot shows the Ignition 15.7 web interface. The top navigation bar includes links for 'USER MANUAL', 'SUPPOR...', 'admin', 'Sign Out', and 'Launch Designer'. The main menu on the left has sections for 'SYSTEMS' (Overview, Performance, Alarm Pipelines, Gateway Scripts, Modules, Redundancy, Reports, SFCs, Tags, Transaction Groups) and 'CONNECTIONS' (Databases, Designers, Devices, Gateway Network, Store & Forward, OPC Connections, OPC-UA Server). The 'Databases' section is currently selected. The central area displays two cards: 'Valid Connections' (1 / 3) and 'Total Throughput' (0.1 queries/sec). Below these are three rows of database connection details:

Name	Driver	Status	Connections	Throughput	Action
Database	MySQL Connector/J	✓ Valid	0 / 8	0.1 queries/sec	<button>Details</button>
MSSQL	Microsoft SQLServer JDBC Driver	⌚ Reconnecting	0 / 8	0.0 queries/sec	<button>Details</button>
MySQL	MySQL Connector/J	⚠ ERROR	0 / 8	0.0 queries/sec	<button>Details</button>

I see error messages and overlays on components that interact with a database.

I think a query is taking a long time to execute.

Troubleshooting Guide - Gateway Network

Loggers

None!

On this page ...

- [Loggers](#)
- [MDC Keys](#)
- [Scenarios](#)

MDC Keys

This list of MDC Keys will turn on specific loggers that relate to the Keys context.

- gan-remote-connection-id
- gan-remote-connection-name

Scenarios

Troubleshooting Guide - Gateway Scripts

Loggers

This is a list of loggers that will be useful in troubleshooting issues with Alarm Notification.

a lot of the loggers had issues, holding off for now.

On this page ...

- [Loggers](#)
- [MDC Keys](#)
- [Scenarios](#)

MDC Keys

This list of MDC Keys will turn on specific loggers that relate to the Keys context.

- script-type
- script-context
- project-name

Scenarios

Troubleshooting Guide - Modules

Loggers

This is a list of loggers that will be useful in troubleshooting issues with Modules.

gateway

gateway.ModuleManager

gateway.ModuleManager.ModuleInfoParser

gateway.ModuleManager.ModuleRoutes

gateway.ModuleManager.ResourceLocator

On this page ...

- [Loggers](#)
 - [gateway](#)
 - [gateway.ModuleManager](#)
 - [gateway.ModuleManager.ModuleInfoParser](#)
 - [gateway.ModuleManager.ModuleRoutes](#)
 - [gateway.ModuleManager.ResourceLocator](#)
- [MDC Keys](#)
- [Scenarios](#)

MDC Keys

This list of MDC Keys will turn on specific loggers that relate to the Keys context.

- module-name

Scenarios

Troubleshooting Guides - Devices

Troubleshooting Guide - Store and Forward

What is a Memory Buffer?

-Explain what the memory buffer is. Explain how the memory buffer directly correlates with Heap. From my experience 100k memory buffer equates to about 3gb of heap.

What is Local Cache and how is it different from Memory Buffer?

-Stored in disk. Where? Explain how there is a directory where Local Cache data is stored per database connection defined in Ignition

-Explain how limitations here are not exactly RAM memory but its more of a disk limitation at this point.

How do the Memory Buffer and Local Cache interact with each other?

-Explain

My Local Cache is full. What do I do?

-Go over how increasing the Local Cache is always an option.

-Talk about how quarantine records count towards Local Cache so if quarantine count equals local cache size, records will be dropped

My Memory Buffer is full. What do I do?

-Pray lol

My Local Cache record count is growing. What can I do?

-Pray lol

-Loggers that would help determine what system is filling up the Local Cache

-You can have two store and forward engines forward data to the same schema in a database concurrently. Would an example of this be a good idea? This is useful in the event that a local cache buffer just keeps growing and we need to have it flush without continuously getting new records added to it.

-Link this back to database troubleshooting, slow queries, contact support.

What does it mean for Store and Forward to "Drop" a record?

-Explain

Discuss the real risk of Store and Forward local cache corruption as it seems someone's local cache corrupts every other day.

-Explain.

Troubleshooting Guide - Tag History

How does Ignition's Tag Historian store data into a database?

-For this section, I would like to see descriptions on how the sqith tables interact.

-Add a small example with screenshots of how a table's _te table entry ties to an entry in the _scinfo table and then the _scinfo table entry relates to a _drv table entry.

-Explain partitions a bit and how their name tells us "when" they were built.

https://docs.google.com/document/d/113WOEECzoownBbsdBFRpBamlH8xP_EkBbBeEiUmi4/edit# This google DOC has a wealth of information that I believe can be condensed for UM use.

How can I check if there is history stored for a specific tag?

-Example on getting tagid for tag and checking _data tables for data for a specific tag.

-Include something helpful so that theyc an convert unixtime to readable timestamp.

How can I check the performance of my Tag Historian?

-Have link here to the Slow Queries page [Slow Queries](#)

-Talk about the _sce table and Stale Data Detection and how this can cause the _sce table to grow at alarming rates ultimately tanking Tag History performance

-How to clear _sce table?

-These are actual benchmark tests that Sales Engineering put out. I am sure we can condense this down to a something we can publish on the UM:

-<https://confluence.ia.local:8443/display/SUP/Historian+Benchmarks+for+MSSQL>

-<https://confluence.ia.local:8443/display/SUP/Historian+Benchmarks+for+MySQL>

-<https://confluence.ia.local:8443/display/SUP/Ignition+History+Storage+Requirements>

-<https://confluence.ia.local:8443/display/SUP/Tag+Historian+Performance+Guide>

How can I improve the performance of my Historian?

-I would like to place a lot of emphasis on consulting a DBA here but I would like a small list of things someone can do Ignition side to try and better performance of a historian.

--Increase DB active/idle connection pool.

--Clear _sce table

--Pre-processed partitions. I was only able to find this in our entire UM for pre-processed partitions and I believe it can be supplemented here:

Enable Pre-processed Partitions	This option is not selected by default. Pre-processed partitions will use more space in the database, but can improve query speed by summarizing data, reducing the amount that must be loaded. Default is false.
Pre-processed Window Size (seconds)	When pre-processing is turned on, the data will be summarized into blocks of this size. Default is 60.

--Decrease query frequency ie query polling, chart polling, etc

--Decreasing storing frequency

Development Server Best Practices

Overview

Many users find a Development Server useful so changes can be tested in a safe environment before being pushed to production. This page details several recommendations and issues to consider when utilizing a Development Server.

Device Considerations

- The Development Server should have its own PLCs. This prevents the Development Server from accidentally writing to a Tag that the Production Server is connected to.
- Device connection names should be the same on both the Development and Production Servers. OPC Tags typically reference the name of the Device Connection in their OPC Item Path, so using the same name on both servers means that Tags can easily be migrated from one server to the other.
- Using PLC simulators is an alternative to the Development Server having its own PLCs. Generally this means finding simulators for your specific devices outside of Ignition.

On this page ...

- Overview
- Device Considerations
- Licensing Considerations
- Database Considerations
- Network Considerations
- Transferring Resources from Development

Licensing Considerations

- The Development Server does not require a licensed Ignition Gateway, but one may be necessary depending on your testing requirements. All modules can run on the [trial](#) for a two-hour testing window, and may be started an unlimited number of times. If you want a more flexible test system for your Development Server, you can add a license to it.

Database Considerations

- The Development Server should have its own [database](#), separate from the production database. This is especially useful when designing screens that manually modify a database table.
- The names of the database connections used by the Development Ignition Gateway should match the names of the connections on the Production Server. Many system functions that interact directly with the database, such as [system.db.runPrepUpdate](#), take the name of the database as a parameter, so using similar names for connections on both servers means you will not have to modify the script once it has been moved to the Production Server.

Network Considerations

- Some users prefer to place their Development Server on a separate [network](#) from the Production Server. This way there can be no confusion as to which server is being accessed. Keep in mind that Clients can only be launched on panels/computers that have network access to the host Gateway, so how this development network is configured will impact where testing clients can be launched.
- If you are restoring a copy of your Production Server on your Development Server, you will need to be cautious: two copies of the same Gateway, on the same network, will both have access to all PLCs and Databases. In this case it is possible for duplicate historical records to be generated, among other potential problems. To prevent this, we recommend keeping the Development Server on a separate network. Alternatively, you could utilize the **Restore Disabled** checkbox when restoring on the Development Server. Once the disabled Development Gateway is up and running, you can edit each connection and point them to testing equipment.

Transferring Resources from Development

- When you are ready to move resources from the Development Server to the Production Server, Gateway Backups should not be utilized as they will include connections to the development resources. Instead, use the [Project Exports](#) and [Tag Exports](#) functionality to transfer resources.
- In cases where there are multiple Ignition Servers (Production or Testing), the [Enterprise Administration Module](#) can make the resource migration easier.

Related Topics ...

- [Connecting to a Device](#)
- [OPC UA Client Connection Settings](#)
- [Simulators](#)

- Licensing and Activation
- Database Connections
- Gateway Network
- Enterprise Administration

Adding Security Certificates into Keystores

Gateway Supplemental Certificates

In some cases you may need to provide additional security certificates so the Gateway can communicate with other systems, such as databases or devices elsewhere on the network. These supplemental certificates can be added to a Gateway by simply placing them in the following directory on the Gateway's file system:

```
%gateway installation directory%data/certificates/supplemental
```

Once added, you will need to restart the Gateway before the certificates will be used.

Supported formats are DER encoded binary X.509, and Base-64 encoded X.509 (PEM-encoded ASCII).

Launchers

You can manually install security certificates to all of our launchers. This is generally only required when the Gateway hosting the application has SSL enabled, and is using either self-signed certificates or certificates signed by a certificate authority internal to a particular organization.

The process is listed below. Note that you will need to repeat the steps below for each applicable Gateway.

1. First, navigate to the Gateway web interface.
2. Download the certificate. Most web browsers provide this functionality, usually via the address bar or some other means.
3. Once you have a copy of the certificate, place the certificate into the following directory:

```
{user folder}\.ignition\clientlauncher-data\certificates
```

When a certificate is placed in the directory above, the launcher will attempt to automatically add the certificate(s) to the local keystore upon application launch.

On this page ...

- [Gateway Supplemental Certificates](#)
- [Launchers](#)

Mass Edits on Large Amount of Tags

I just upgraded the firmware on my Allen Bradley PLC and how all my tags are broken

-This is a tricky one but I believe it should be somewhere in the manual. Just a simple example where we export, Find/Replace with Notepad++ and re-import.

-The most common example of this is when someone updates their firmware on the AB PLC and the OPC Item Path syntax changes

Mass tag edits were easy in 7.9 but are hard in 8.0. Why?

-Differences in how mass edits on tags changed between 7.9 and 8.0. Explain the tag overhaul and how tags are a lot faster now in 8.0.

-Support has recommended in the past that mass tag edits be performed via scripting. Why not have a small example?

Storing image in database as blob

How can I store an image in a database?

-Example

Ignition Advanced Configurations

How to allocate more heap to Ignition?

-Example.

How can a logging level be set on service start?

-Example on configuring the gateway.xml file to set logging levels on service start.

How to increase the size and count of wrapper.log files?

-Example.

Add more as I think of them.

Setting Vision Application to Launch When Server Boots Up

- Window start up folder
- Mac and linux alternatives????