Scripting Functions

The Ignition scripting API, which is available under the module name "system", is full of functions that are useful when designing projects in Ignition. From running database queries, manipulating components, to exporting data, scripting functions can help. Some of these functions only work in the Gateway scope, and other only work in the Client scope, while the rest will work in any scope.

Additional information on scripting Ignition can be found in the Scripting section.

In this section, we cover all of the built in scripting functions available inside of Ignition. Each page will have a banner at the top that looks like this:

System Library

Watch the Video

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

This lets you know that you are looking at a function for the Python scripting language.

Keyboard Shortcut

A complete list of these functions (with their definitions) is available wherever you can add a script. Just type **system.** and then press **Ctrl+Space** to get a list of all the functions available. If you keep typing, the list will even be automatically narrowed down for you!

System Functions

You can see below there are many different categories of system functions available for your use. For an overview and syntax scripting, refer to the Python Scripting section.

system.alarm

system.bacnet

system.dataset

system.date

system.db

system.device

system.dnp3

system.eam

system.file

system.groups

system.gui

system.math

system.nav

system.net

system.opc

system.opchda

system.opcua

system.perspective

system.print

system.roster

system.secsgem

system.security

system.serial

system.sfc

system.tag

system.twilio

system.user

system.util

system.project

system.report

system.alarm

Alarm Functions

The following functions give you access to view and interact with the Alarm system in Ignition.

In This Section ...

system.alarm.acknowledge

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

Description

Acknowledges any number of alarms, specified by their event ids. The event id is generated for an alarm when it becomes active, and is used to identify a particular event from other events for the same source. The alarms will be acknowledged by the logged in user making the call. Additionally, acknowledgement notes may be included and will be stored along with the acknowledgement.

This function uses different parameters based on the scope of the script calling it. Both versions are listed below.

Client Permission Restrictions

Permission Type: Alarm 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 - Client Scripts

system.alarm.acknowledge(alarmIds, notes)

- Parameters

  String[] **alarmIds** - List of alarm event ids (uuids) to acknowledge.

  String **notes** - A string that will be used as the Ack Note on each acknowledged alarm event. If set to `None` , then an Ack Note note will not be assigned to the alarm event.

- Returns

  Nothing

- Scope

  Vision Client

Syntax - Gateway Scripts

system.alarm.acknowledge(alarmIds, notes, username)

- Parameters

  String[] **alarmIds** - List of alarm event ids (uuids) to acknowledge.

  String **notes** - A string that will be used as the Ack Note on each acknowledged alarm event. If set to `None` , then an Ack Note note will not be assigned to the alarm event.

  String **username** - The user that acknowledged the alarm. **NOTE**: This parameter is only used when called from a gateway scoped script. This parameter should be omitted from any client-based scripts.

- Returns

  Nothing

- Scope

  Gateway, Perspective Session

Examples

Code Snippet - Acknowledging an Alarm in Client Scope

```
# This example shows the basic syntax for acknowledging an alarm from a client-based script
system.alarm.acknowledge(['c27c06d8-698f-4814-af89-3c22944f58c5'],'Saw this alarm, did
something about it.')
```

Code Snippet - Acknowledging an Alarm in Gateway Scope

```
# This example shows the basic syntax for acknowledging an alarm from a gateway-based script
system.alarm.acknowledge(['c27c06d8-698f-4814-af89-3c22944f58c5'],'Saw this alarm, did
something about it.', 'admin')
```

Code Snippet - Acknowledging Selected Alarms from a Table

```
# This code snippet could be used as a mouseReleased event handler on a Table component (not
an Alarm Status Table component)
# whose data was the return value of the system.alarm.queryStatus  function.
# It presents a right-click menu to acknowledge the currently selected alarms (for more than
one, the table must be set to allow multiple selection).
# This example does not ask for an ack message, and therefore might fail if the alarms we're
attempting to acknowledge require notes.
# Also, note that the system will ignore any alarms that have already been acknowledged.

if event.button==3:
   rows = event.source.selectedRows
   data = event.source.data
   if len(rows)>0:
      uuids = [str(data.getValueAt(r,'EventId')) for r in rows]
      def ack(event, uuids=uuids):
         import system
         system.alarm.acknowledge(uuids, None)
      menu = system.gui.createPopupMenu({'Acknowledge':ack})
      menu.show(event)
```

Keywords

system alarm acknowledge, alarm.acknowledge

system.alarm.cancel

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

Description

Cancels any number of alarms, specified by their event ids. The event id is generated for an alarm when it becomes active, and is used to identify a particular event from other events for the same source. The alarm will still be active, but will drop out of alarm pipelines.

Client Permission Restrictions

Permission Type: Alarm 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.alarm.cancel(alarmIds)

- Parameters

  String[] alarmIds - List of alarm event ids (uuids) to cancel.

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Cancelling an Alarm

```
# This example shows the basic syntax for cancelling an alarm.

system.alarm.cancel(['c27c06d8-698f-4814-af89-3c22944f58c5'])
```

Code Snippet - Cancelling All Currently Active Alarms

```
# To cancel all currently active alarms:

ids = []
results = system.alarm.queryStatus(state=["ActiveUnacked", "ActiveAcked"])
for result in results:
 id = result.getId()
 ids.append(str(id))

system.alarm.cancel(ids)
```

Keywords

system alarm cancel, alarm.cancel

system.alarm.createRoster

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

Description

This function creates a new roster. Users may be added to the roster through the Gateway or the Roster Management component

Client Permission Restrictions

Permission Type: Alarm 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.alarm.createRoster(name, description)

- Parameters

  String name - The name for the new roster

  String description - A description for the new roster. Required, but can be blank.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Creating a New Roster

```
# This example creates a new roster
name = 'MyRoster'
description = 'A roster created by scripting'
system.alarm.createRoster(name, description)
```

Keywords

system alarm createRoster, alarm.createRoster

system.alarm.getRosters

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

Description

This function returns a mapping of roster names to a list of usernames contained in the roster.

Client Permission Restrictions

This scripting function has no Client Permission or Perspective Session restrictions.

Syntax

system.alarm.getRosters()

- Parameters

  None

- Returns

  PyDict - 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, Vision Client, Perspective Session

Code Examples

Code In Action - Listing All the Users in a Roster

```
# This script will get all the rosters and list the users in them.
rosters = system.alarm.getRosters()
for key, values in rosters.iteritems():
        # key is the roster name, values is a dict of usernames
        print 'Roster', key, 'contains these users:'
        for value in values:
                print '  ', value
```

Code In Action - Listing All the Users in a Roster from a Perspective Session

```
# This script will get all the rosters and list the users in them.
rosters = system.alarm.getRosters()
for key, values in rosters.iteritems():
        # key is the roster name, values is a dict of usernames
        system.perspective.print('Roster ' + key + ' contains these users:')
        for value in values:
                system.perspective.print('  ' + value)
```

Output

```
Roster Admins contains these users:
   admin
Roster Supervisors contains these users:
   asmith
   jdoe
```

Keywords

system alarm getRosters, alarm.getRosters

system.alarm.getShelvedPaths

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

Description

Returns a list of ShelvedPath objects, which each represent a shelved alarm.

Client Permission Restrictions

This scripting function has no Client Permission or Perspective Session restrictions..

Syntax

system.alarm.getShelvedPaths()

- Parameters

  Nothing

- Returns

  List - A list of ShelvedPath objects. ShelvedPath objects can be examined with getExpiration, getHitCount, getPath, getShelveTime, getUser, and isExpired.

- Scope

  Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Getting Paths for All Shelved Alarms

```
# The following code prints a list of the shelved alarms paths and prints them to the console.
paths = system.alarm.getShelvedPaths()
for p in paths:
   print "Path: %s, Shelved by: %s, expires: %s, is expired? %s" % (p.getPath(), p.getUser(),
p.getExpiration(), p.isExpired())
```

Keywords

system alarm getShelvedPaths, alarm.getShelvedPaths

system.alarm.listPipelines

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

Description

Will return a list of the available Alarm Notification Pipelines in a project.

The legacy behavior of this function (7.9 and prior) did not have any parameters, and would always check all projects for pipelines. See the Upgrade Guide for more details.

Client Permission Restrictions

This scripting function has no Client Permission or Perspective Session restrictions.

Syntax

system.alarm.listPipelines([projectName])

- Parameters

  String projectName - The project to check alarm pipelines for. If omitted, will look for a project named "alarm-pipelines".

- Returns

  List - A list of pipeline names. The list may be empty if no pipelines exist. Unsaved name changes will not be reflected in the list.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Listing the Alarm Pipelines in a Project

```
# This script will print a list of all alarm pipeline names in the “alarm-pipelines” project.
pipelines = system.alarm.listPipelines()
for pipeline in pipelines:
        print pipeline
```

Output

```
Emergency_Pipeline
Test
SMS_Pipeline
```

Keywords

system alarm listPipelines, alarm.listPipelines

system.alarm.queryJournal

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

Description
Queries the specified journal for historical alarm events. The result is a list of alarm events, which can be queried for individual properties. The result object also has a getDataset() function that can be used to convert the query results into a normal dataset, with the columns: EventId, Source, DisplayPath, EventTime, EventState, Priority, IsSystemEvent Click here for more information on alarm properties

Client Permission Restrictions
This scripting function has no Client Permission or Perspective Session restrictions.

Syntax

system.alarm.queryJournal(startDate, endDate, journalName, priority, state, path, source, displaypath, all_properties, any_properties, defined, includeData, includeSystem, isSystem)

- Parameters

  Date startDate - The start of the time range to query. Defaults to 8 hours previous to now if omitted. Time range is inclusive.

  Date endDate - The end of the time range to query. Defaults to "now" if omitted.

  String journalName - The journal name to query.

  String[] priority - A list of possible priorities to match. Priorities can be specified by name or number, with the values: Diagnostic(0), Low(1), Medium(2), High(3), Critical(4).

  String[] state - A list of the event state types to match. Valid values are "ClearUnacked", "ClearAcked", "ActiveUnacked", and "ActiveAcked".

  String[] path - A list of possible source paths to search at. The wildcard "*" may be used.

  String[] source - A list of possible source paths to search at. The wildcard "*" may be used.

  String[] displaypath - A list of display paths to search at. Display paths are separated by "/", and if a path ends in " /*", everything below that path will be searched as well.

  Object[][] all_properties - A set of property conditions, all of which must be met for the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=","!=","<","<="," >",">=". Only the first two conditions may be used for string values.

  Object[][] any_properties - A set of property conditions, any of which will cause the overall the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=","!=","<"," <=",">",">=". Only the first two conditions may be used for string values.

  String[] defined - A list of string property names, all of which must be present on an event for it to pass.

  Boolean includeData - Whether or not event data should be included in the return. If this parameter is false, and if there are no conditions specified on associated data, the properties table will not be queried.

  Boolean includeSystem - Specifies whether system events are included in the return.

  Boolean isSystem - Specifies whether the returned event must or must not be a system event.

- Returns

  AlarmQueryResult - The AlarmQueryResult object is functionally a list of AlarmEvent objects. The AlarmQueryResult object has a built-in getDataset() function that will return a Standard Dataset containing the Event Id (UUID of the alarm), Source Path, Display Path, Event Time, State (as an integer), and Priority (as an integer).

  Additionally, each AlarmEvent inside of the AlarmQueryResult object has several built-in methods to extract alarm information. More details on these methods can be found on the Alarm Event Properties Reference page.

Important

Each item in the resulting list is a separate alarm event: an alarm becoming active is one item, while the same alarm becoming acknowledged is a separate item. This differs from system.alarm.queryStatus() which groups each event into a single item.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Querying the Alarm Journal

```
# This example shows the basic syntax for querying from the journal in a button's
actionPerformed event, with a date range selector ("Range"), storing the results back to a
table called "Table":

table = event.source.parent.getComponent("Table")
range= event.source.parent.getComponent("Range")

results = system.alarm.queryJournal(journalName="Journal", startDate=range.startDate,
endDate=range.endDate)
table.data = results.getDataset()
```

Code Snippet - Querying the Alarm Journal With Filters

```
# This example extends the previous to only include non-acknowledged events of High or
Critical severity, who have associated data called "Department", set to "maintenance". It
also excludes system events (shelving notifications, etc):

table = event.source.parent.getComponent("Table")
range= event.source.parent.getComponent("Range")

results = system.alarm.queryJournal(journalName="Journal", startDate=range.startDate,
endDate=range.endDate, state=['ActiveUnacked', 'ClearUnacked'], all_properties=
[("Department","=","maintenance")], priority=["High", "Critical"], includeSystem=False)
table.data = results.getDataset()
```

Keywords

system alarm queryJournal, alarm.queryJournal

system.alarm.queryStatus

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

Description
Queries the current state of alarms. The result is a list of alarm events, which can be queried for individual properties. The result object also has a getDataset() function that can be used to convert the query results into a normal dataset, with the columns: EventId, Source, DisplayPath, EventTime, State, Priority **Note:** Depending on the number of alarm events in the system, this function can be fairly intensive and take a while to finish executing. Which can be problematic if the application is attempting to show the results on a component (such as using this function to retrieve a count of alarms). In these cases it's preferred to call this function in a gateway script of some sort (such as a timer script), and store the results in a tag, which can easily be accessed by a component binding. The Tag Properties page contains more information on Alarm Properties

Client Permission Restrictions
This scripting function has no Client Permission or Perspective Session restrictions.

Syntax

system.alarm.queryStatus(priority, state, path, source, displaypath, all_properties, any_properties, defined, includeShelved, provider)

- Parameters

  String[] priority - A list of possible priorities to match. Priorities can be specified by name or number, with the values: Diagnostic(0), Low(1), Medium(2), High(3), Critical(4).

  String[] state - A list of states to allow. Valid values: "ClearUnacked", "ClearAcked", "ActiveUnacked", "ActiveAcked".

  String[] path - A list of possible source paths to search at. The wildcard "*" may be used. Works the same as the source argument, and either can be used.

  String[] source - A list of possible source paths to search at. The wildcard "*" may be used. Works the same as the path argument, and either can be used.

  String[] displaypath - A list of display paths to search at. Display paths are separated by "/", and if a path ends in "/*", everything below that path will be searched as well.

  Object[][] all_properties - A set of property conditions, all of which must be met for the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=","!=","<","<=",">",">=". Only the first two conditions may be used for string values.

  Object[][] any_properties - A set of property conditions, any of which will cause the overall the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=","!=","<","<=",">",">=". Only the first two conditions may be used for string values.

  String[] defined - A list of string property names, all of which must be present on an event for it to pass.

  Boolean includeShelved - A flag indicating whether shelved events should be included in the results. Defaults to "false".

  List[String] provider - A list of tag providers to include in the query. [optional]

- Returns

  AlarmQueryResult - The AlarmQueryResult object is functionally a list of AlarmEvent objects. The AlarmQueryResult object has a built-in **getDataset()** function that will return a Standard Dataset containing the Event Id (UUID of the alarm), Source Path, Display Path, Event Time, State (as an integer), and Priority (as an integer).

  Additionally, each AlarmEvent inside of the AlarmQueryResult object has several built-in methods to extract alarm information. More details on these methods can be found on the Alarm Event Properties Reference page.

Important

Each item in the resulting list is a combination of each alarm event for the same alarm: details for when the alarm became active, acknowledged, and cleared are combined into a single item. This differs from system.alarm.queryJournal() which splits these events into separate items.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Querying Alarm Status

```
# This example queries the state of all tags named "HiAlarm", and puts the results in a
Vision table Component named "Table" (this assumes it's being run from a button on the same
screen)
#   Note that this example is simple for the sake of brevity. Normally you'll want to use
system.util.invokeAsynchronous to search for alarms in a separate thread, especially so if
calling
#   this function from a component based script. See the next example for more information.

table = event.source.parent.getComponent("Table")

results = system.alarm.queryStatus(source=["*HiAlarm*"])
table.data = results.getDataset()
```

Code Snippet - Call queryStatus in a Separate Thread

```
# In this example we'll call system.alarm.queryStatus in a separate thread, and return the
results to the data property on a Vision Table component. Similar to the example above.
#   What makes this example different is that it offers better performance when calling from
a Vision component. Depending on the number of alarm events in the system, queryStatus
#   may take a significant amount of time to finish, which would lock up a Vision Client
while the script is running in the GUI thread. Thus this example will use
#   system.util.invokeAsynchronous to call queryStatus in a separate thread, and then system.
util.invokeLater make any changes to our components.


# Define a function that will retrieve alarm data in a separate thread
def getAlarms():
        # Call queryStatus to retrieve the alarm data we're looking for, and store the
results in a variable.
        #   In this case, we're looking for alarm events that contain the word "Sensor" in
the source path.
        results = system.alarm.queryStatus(source=["*Sensor*"])

        # From this same script, define a separate function that will later interact with the
        #    GUI, allowing us to move our alarm data over to a component
        # We're also using the getDataset() function on the object returned by queryStatus,
        #   since that will provide a dataset that our table component will expect.
        def setTheTable(alarms = results.getDataset()):

                # Replace the property reference below with a path leading to whichever
property
                #    you want to move the alarm data to.
                event.source.parent.getComponent("Table").data = alarms

        # The last thing we'll do in the separate thread is call invokeLater
        #   which will let our setTheTable function run in the GUI thread
        system.util.invokeLater(setTheTable)


# Call the getAlarms function in a separate thread, which starts the whole
process
system.util.invokeAsynchronous(getAlarms)
```

Code Snippet - Querying Alarm Status Using any_properties

```
# The any_properties parameter allows you to filter the results for specific properties. This
is useful when searching for alarms that contain associated data.

# Build a List of Tuples that represent the properties to search for. In this case, if our
alarms have an Associated Data named 'Group', we can use
# the following to search for potential values
props = [("Group", "=", "value1"), ("Group", "=", "value2")]
state = ["ActiveUnacked", "ActiveAcked"]

alarms = system.alarm.queryStatus(any_properties = props, state = state)

# Here we're printing out the number of alarms that meet our criteria. We could replace this
and further examine each individual alarm in a for-loop instead.
print len(alarms)
```

Keywords
system alarm queryStatus, alarm.queryStatus

system.alarm.shelve

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

Description

This function shelves the specified alarms for the specified amount of time. The paths may be either source paths, or display paths. The time can be specified in minutes (timeoutMinutes) or seconds (timeoutSeconds). If an alarm is already shelved, this will overwrite the remaining time. To unshelve alarms, this function may be used with a time of "0".

Client Permission Restrictions

Permission Type: Alarm 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.alarm.shelve(path, timeoutSeconds, timeoutMinutes)

- Parameters

  String[] path - A list of possible source paths to search at. If a path ends in "/*", the results will include anything below that path.

  Integer timeoutSeconds - The amount of time to shelve the matching alarms for, specified in seconds. 0 indicates that matching alarm events should now be allowed to pass.

  Integer timeoutMinutes - The amount of time to shelve the matching alarms for, specified in minutes. 0 indicates that matching alarm events should now be allowed to pass.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Shelving Alarms

```
# This example assumes that data has been loaded into a table ("Table") from system.alarm.
queryStatus, and it shelves the selected alarms for 5 minutes.
# It also assumes that it is being executed from a button's actionPerformed event.

table = event.source.parent.getComponent('Table')
rows = table.selectedRows
data = table.data
if len(rows)>0:
   sourcePaths = [str(data.getValueAt(r,'Source')) for r in rows]
   system.alarm.shelve(path=sourcePaths,timeoutMinutes=5)
```

Keywords

system alarm shelve, alarm.shelve

system.alarm.unshelve

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

Client Permission Restrictions

Permission Type: Alarm 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.alarm.unshelve(path)

- Parameters

  String[] path - A list of possible source paths to search at. If a path ends in "/*", the results will include anything below that path.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are not code examples for this function.

Keywords

system alarm unshelve, alarm.unshelve

system.bacnet

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

Functions

The following functions

In This Section ...

system.bacnet.synchronizeTime

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

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

Description

Notifies the remote device of the correct current time, which is the system time (factoring in timezone and DST) of the server Ignition is running on.

To use this function, the BACnet driver must be installed.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.bacnet.synchronizeTime(deviceName)

- Parameters

  String deviceName- The name of the configured BACnet/IP device instance to write from.

- Returns

  nothing

- Scope

  Gateway

Code Examples

There are no examples associated with this scripting function.

Keywords

system bacnet synchronizeTime, bacnet.synchronizeTime

system.bacnet.synchronizeTimeUtc

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

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

Description

Notifies the remote device of the correct current time in UTC.

To use this function, the BACnet driver must be installed.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.bacnet.synchronizeTimeUtc(deviceName)

- Parameters

  String deviceName - The name of the configured BACnet/IP device instance to write from.

- Returns

  nothing

- Scope

  Gateway

Code Examples

There are no examples associated with this scripting function.

Keywords

system bacnet synchronizeTimeUtc, bacnet.synchronizeTimeUtc

system.bacnet.writeWithPriority

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

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

Description

Write to the Present_Value attribute of an object with a custom priority level.

⚠ To use this function, the BACnet driver must be installed.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.bacnet.writeWithPriority(deviceName, objectType, objectId, value, priority)

- Parameters

  String deviceName - The name of the configured BACnet/IP device instance to write from.

  Integer objectType - The numeric id of the objectType of the object instance being written to. See the objectType reference table below.

  Integer objectId - The object instance number to write to.

  Object value - The value to write. Clearing a value can be accomplished by writing a None value.

  Integer priority - The priority level to write the value at. Must be in the range [1...16].

- Returns

  nothing

- Scope

  Gateway

objectType Reference

Object	ID
Analog Input	0
Analog Output	1
Analog Value	2
Binary Input	3
Binary Output	4
Binary Value	5
Device	8
Large Analog Value	46
Multi-State Input	13
Multi-State Output	14
Multi-State Value	15

Code Examples

Example 1

```
# Write a value of 'True' to Binary Value 1 with a Priority of 7

deviceName = 'BACnet Remote'
objectType = 5 # Binary Value
objectId = 1
value = True
priority = 7


system.bacnet.writeWithPriority(deviceName, objectType, objectId, value, priority)
```

Example 2

```
# Values can be cleared by writing a None value.

deviceName = 'BACnet Remote'
objectType = 5
objectId = 1
value = None
priority = 7

system.bacnet.writeWithPriority(deviceName, objectType, objectId, value, priority)
```

Keywords

system bacnet writeWithPriority, bacnet.writeWithPriority

system.dataset

Dataset Functions

The following functions give you access to view and interact with datasets.

In This Section ...

system.dataset.addColumn

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

Description

Takes a dataset and returns a new dataset with a new column added or inserted into it.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.addColumn(dataset [, colIndex], col, colName, colType)

- Parameters

  Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

  int colIndex - The index (starting at 0) at which to insert the new column. Will throw an IndexError if less than zero or greater than the length of the dataset. If omitted, the new column will be appended to the end. [optional]

  PySequence col - A Python sequence representing the data for the new column. Its length must equal the number of rows in the dataset.

  String colName - The name of the column.

  PyType colType - The type of the of the column. The type can be the Python equivalent of String, Long, Double, Short, Integer, Float, Boolean, null, or java.util.Date if they exist.

- Returns

  Dataset - A new dataset with the new column inserted or appended.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will work on a Button component on a Vision window, given 2 Vision bar charts
with default values.
# The script takes the dataset from Bar Chart 1, adds a column of integers called Center Area
to the end of the existing data,
# and displays the new dataset in Bar Chart 2.

ds1 = event.source.parent.getComponent('Bar Chart 1').data
colCount = ds1.getColumnCount()
columnName = "Center Area"
columnData = []
for i in range(ds1.getRowCount()):
 # Append sample data to row
 columnData.append(i * 10)

ds2 = system.dataset.addColumn(ds1, colCount, columnData, columnName, int)
event.source.parent.getComponent('Bar Chart 2').data = ds2
```

Keywords

system dataset addColumn, dataset.addColumn

system.dataset.addRow

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

Description

Takes a dataset and returns a new dataset with a new row added or inserted into it.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.addRow(dataset [, rowIndex], row)

- Parameters

  Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

  int rowIndex - The index (starting at 0) at which to insert the new row. Will throw an IndexError if less than zero or greater than the length of the dataset. If omitted, the new row will be appended to the end. [optional]

  PySequence row - A Python sequence representing the data for the new row. Its length must equal the number of columns in the dataset.

- Returns

  Dataset - A new dataset with the new row inserted or appended.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would add a new option to a Vision Dropdown List by adding a row to its
underlying dataset.
# Notice how the last line assigns the return value of the addRow function to the dropdown's
data property.

# This script should be on a button that is a sibling to a Dropdown List component named
"Dropdown".
dropdown = event.source.parent.getComponent("Dropdown")
newRow = [5, "New Option"]
dropdown.data = system.dataset.addRow(dropdown.data, newRow)
```

Code Snippet

```
# This snippet would add a new option into a Dropdown component just like above, but at the
beginning:
dropdown = event.source.parent.getComponent("Dropdown")
newRow = [5, "New Option"]
dropdown.data = system.dataset.addRow(dropdown.data, 0, newRow)
```

Keywords

system dataset addRow, dataset.addRow

system.dataset.addRows

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

Description

Takes a dataset and returns a new dataset with new rows added or inserted into it.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.addRows(dataset [, rowIndex], rows)

- Parameters

  Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

  int rowIndex - The index (starting at 0) at which to insert the new row. Will throw an IndexError if less than zero or greater than the length of the dataset. If omitted, the new row will be appended to the end. [optional]

  PySequence rows - A Python sequence of sequences representing the data for the new rows. The length of each sequence must equal the number of columns in the dataset.

- Returns

  Dataset - A new dataset with the new rows inserted or appended.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would add new options to a Vision Dropdown List by adding a row to its
underlying dataset.
# Note that the last line assigns the return value of the addRows function to the dropdown's
data property.

dropdown = event.source.parent.getComponent("Dropdown")
newRow = [[5, "New Option"], [6, "Another New Option"]]
dropdown.data = system.dataset.addRows(dropdown.data, newRow)
```

Code Snippet

```
# This snippet would add new options into a Dropdown component just like above, but at the
beginning:

dropdown = event.source.parent.getComponent("Dropdown")
newRow = [[5, "New Option"], [6, "Another New Option"]]
dropdown.data = system.dataset.addRows(dropdown.data, 0, newRow)
```

Keywords

system dataset addRows, dataset.addRows

system.dataset.appendDataset

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

Description

Takes two different datasets and returns a new dataset with the second dataset appended to the first. Will throw an error if the number and types of columns do not match.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.appendDataset(dataset1, dataset2)

- Parameters

  Dataset dataset - The dataset that will come first in the returned dataset.

  Dataset dataset - The second dataset that will be appended to the end in the returned dataset.

- Returns

  Dataset - A new dataset that is a combination of the original two datasets.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example pulls in the datasets from two different Vision table components, combines
them, then writes the results to a third table.
# The two source datasets must have the same number and types of columns.

data1 = event.source.parent.getComponent('Table 1').data
data2 = event.source.parent.getComponent('Table 2').data
comboData = system.dataset.appendDataset(data1, data2)
event.source.parent.getComponent('Table 3').data = comboData
```

Keywords

system dataset appendDataset, dataset.appendDataset

system.dataset.clearDataset

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

Description

Takes a dataset and returns a new dataset with all of the same column names, but all of the rows deleted.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.clearDataset(dataset)

- Parameters

  Dataset dataset - The starting dataset. If NULL, a NULL dataset will be returned.

- Returns

  Dataset - A new dataset with no data.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example pulls in the dataset from a Vision Table component, clears it, then writes the
empty dataset back to the table.

data = event.source.parent.getComponent('Table').data
event.source.parent.getComponent('Table').data = system.dataset.clearDataset(data)
```

Keywords

system dataset clearDataset, dataset.clearDataset

system.dataset.dataSetToHTML

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

Description

Formats the contents of a dataset as an HTML page, returning the results as a string. Uses the <table> element to create a data table page.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.dataSetToHTML(showHeaders, dataset, title)

- Parameters

  boolean showHeaders - If true(1), the HTML table will include a header row.

  Dataset dataset - The dataset to export

  String title - The title for the HTML page.

- Returns

  String - The HTML page as a string.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet would run a SQL query against a database, and turn the results into a string
containing HTML. It then writes the string to a file on the local hard drive.

results = system.db.runNamedQuery("Fetch Records",{})
html = system.dataset.dataSetToHTML(1, results, "Production Report")
filePath = "C:\\output\\results.html"
system.file.writeFile(filePath, html)
```

Keywords

system dataset dataSetToHTML, dataset.dataSetToHTML

system.dataset.deleteRow

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

Description

Takes a dataset and returns a new dataset with a row removed.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.deleteRow(dataset, rowIndex)

- Parameters

  Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

  int rowIndex - The index (starting at 0) of the row to delete. Will throw an IndexError if less than zero or greater than the row count of the dataset -1.

- Returns

  Dataset - A new dataset with the specified row removed.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would remove the selected row from a Vision List component, by re-assigning
the List's data property to the new dataset returned by the deleteRow function.

myList = event.source.parent.getComponent("List")
row = myList.selectedIndex
if row != -1: # make sure there is something selected
   myList.data = system.dataset.deleteRow(myList.data, row)
```

Keywords

system dataset deleteRow, dataset.deleteRow

system.dataset.deleteRows

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

Description

Takes a dataset and returns a new dataset with one or more rows removed.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.deleteRows(dataset, rowIndices)

- Parameters

  Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

  int[] rowIndices - The indices (starting at 0) of the rows to delete. Will throw an IndexError if any element is less than zero or greater than the number of rows in the dataset - 1.

- Returns

  Dataset - A new dataset with the specified rows removed.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would remove several rows from a Vision Table component, by re-assigning the
Table's data property to the new dataset returned by the deleteRows function.

ds = event.source.parent.getComponent('Table').data
rows = [0,2,3,4]
ds = system.dataset.deleteRows(ds, rows)
event.source.parent.getComponent('Table').data = ds
```

Keywords

system dataset deleteRows, dataset.deleteRows

system.dataset.exportCSV

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

Description

Exports the contents of a dataset as a CSV file, prompting the user to save the file to disk. Note that, to write silently to a file, you cannot use the dataset.export* functions. Instead, use the toCSV() function.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.exportCSV(filename, showHeaders, dataset)

- Parameters

  String filename - A suggested filename to save as.

  boolean showHeaders - If true (1), the CSV file will include a header row.

  Dataset dataset - The dataset to export.

- Returns

  String - The path to the saved file, or None if the action was canceled by the user.

- Scope

  Vision Client

Code Examples

Code Snippet

```
#This snippet would prompt the user to save the data currently displayed in a Table component
to a CSV file, and would open the file (in an external program, presumably Excel) after a
successful save.

table = event.source.parent.getComponent("Table")
filePath = system.dataset.exportCSV("data.csv", 1, table.data)
if filePath != None:
        system.net.openURL("file:///"+filePath.replace('\\','/'))
```

Keywords

system dataset exportCSV, dataset.exportCSV

system.dataset.exportExcel

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

Description

Exports the contents of a dataset as an Excel spreadsheet, prompting the user to save the file to disk. Uses the same format as the dataSetToExcel function. Note that, to write silently to a file, you cannot use the dataset.export* functions. Instead, use the toExcel() function.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.exportExcel(filename, showHeaders, dataset[, nullsEmpty])

- Parameters

  String filename - A suggested filename to save as.

  boolean showHeaders - If true (1), the spreadsheet will include a header row.

  Object[] dataset - A sequence of datasets, one for each sheet in the resulting workbook.

  Boolean nullsEmpty - If True (1), the spreadsheet will leave cells with NULL values empty, instead of allowing Excel to provide a default value like 0. Defaults to False. [Optional]

- Returns

  String - The path to the saved file, or None if the action was canceled by the user.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This snippet would prompt the user to save the data currently displayed in a Table
component to an Excel-compatible spreadsheet file, and would open the file after a successful
save.

table = event.source.parent.getComponent("Table")
filePath = system.dataset.exportExcel("data.xls", 1, table.data)
if filePath != None:
   system.net.openURL("file://"+filePath)
```

Keywords

system dataset exportExcel, dataset.exportExcel

system.dataset.exportHTML

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

Description

Exports the contents of a dataset to an HTML page. Prompts the user to save the file to disk.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.exportHTML(filename, showHeaders, dataset, title)

- Parameters

  String filename - A suggested filename to save as.

  boolean showHeaders - If true (1), the HTML table will include a header row.

  Dataset dataset - The dataset to export.

  String title - The title for the HTML page.

- Returns

  String - The path to the saved file, or None if the action was canceled by the user.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This snippet would prompt the user to save the data currently displayed in a Vision Table
component to an HTML file, and would open the file in the default web  browser after a
successful save.

table = event.source.parent.getComponent("Table")
filePath = system.dataset.exportHTML("data.html", 1, table.data, "Production Report")
if filePath != None:
   system.net.openURL("file://"+filePath)
```

Keywords

system dataset exportHTML, dataset.exportHTML

system.dataset.filterColumns

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

Description

Takes a dataset and returns a view of the dataset containing only the columns found within the given list of columns.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.filterColumns(dataset, columns)

- Parameters

  Dataset dataset - The starting dataset.

  PySequence columns - A list of columns to keep in the returned dataset. The columns may be in integer index form (starting at 0), or the name of the columns as Strings.

- Returns

  Dataset - A new dataset containing the filtered columns. The order of columns in this dataset is determined by the column order provided to the columns parameter.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example takes the dataset from a four column Bar Chart and displays a subset of the
data in two separate tables.  This is performed in a button component actionPerformed script.

chartData = event.source.parent.getComponent('Bar Chart').data

northSouth = [0, 1] # North Area, South Area cols
eastWest = ["East Area", "West Area"]

filteredData = system.dataset.filterColumns(chartData, northSouth)
event.source.parent.getComponent('NorthSouthTable').data = filteredData

filteredData = system.dataset.filterColumns(chartData, eastWest)
event.source.parent.getComponent('EastWestTable').data = filteredData
```

Keywords

system dataset filterColumns, dataset.filterColumns

system.dataset.formatDates

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

Description

Returns a new dataset with Date columns as strings formatted according to the dateFormat specified.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.formatDates(dataset, dateFormat, locale)

- Parameters

  Dataset dataset - The starting dataset to format.

  String dateFormat - A valid Java DateFormat string, representing how the date should be formatted. For example: "yyyy-MM-dd HH:mm:ss". See system.date.format for more information on the valid characters.

  Locale locale- The Locale to use for formatting. The Locale object is actually any of Java's Locales, which can be found here. The java Locale library must be imported, and the Locale must be defined in all caps. See the second example below for an idea of how that works. If no Locale is specified, the default Locale will be used. [optional]

- Returns

  Dataset - A new dataset, containing the formatted dates.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example takes the dataset from a table component and formats the dates to look like
Fri, Jan 22, 2018.

data = event.source.parent.getComponent('Table').data
formattedData = system.dataset.formatDates(data, "EEE, MMM d, yyyy")
```

Code Snippet

```
# This example formats the date similarly to the last example, but uses the Italian Locale,
which causes the dates to be formatted with the Locale.

from java.util import Locale

data = event.source.parent.getComponent('Table').data
locale = Locale.ITALY
formattedDataFr = system.dataset.formatDates(data, "yyyy-MM-dd HH:mm:ss", locale)
```

Keywords

system dataset formatDates, dataset.formatDates

system.dataset.fromCSV

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

Description

Converts a dataset stored in a CSV formatted string to a dataset that can be immediately assignable to a dataset property in your project. Usually this is used in conjunction with system.file.readFileAsString when reading in a CSV file that was exported using system.dataset.toCSV . The CSV string must be formatted in a specific way:

```
#NAMES
Col 1,Col 2,Col 3
#TYPES
I,str,D
#ROWS,6
44,Test Row 2,1.8713151369491254
86,Test Row 3,97.4913421614675
0,Test Row 8,20.39722542161364
78,Test Row 9,34.57127071614745
20,Test Row 10,76.41114659745085
21,Test Row 13,13.880548366871926
```

The first line must be #NAMES

The second line must list the names of the columns of the dataset separated by commas

The third line must be #TYPES

The fourth line must list the type of each column of the dataset in order, separated by commas

Integer = I

String = str

Double = D

Date = date

Long = L

Short = S

Float = F

Boolean = B

The fifth line must be #ROWS followed by a comma and then the number of rows of data (i.e. #ROWS, 6)

The following lines will be your data, each column value separated by a comma; each row on a separate line. The number of rows must match what was specified on line 5

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.fromCSV(csv)

- Parameters

  String csv - A string holding a CSV dataset.

- Returns

  Dataset - A new dataset.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# In this example it is assumed that the CSV file being read was a dataset that was
previously exported using system.dataset.toCSV:
# Specify file path
file_path = "C:\\my_dataset.csv"
# Read in the file as a string
data_string = system.file.readFileAsString(file_path)
# Convert the string to a dataset and store in a variable
data = system.dataset.fromCSV(data_string)
# Assign the dataset to a table
event.source.parent.getComponent('Table').data = data
```

Keywords

system dataset fromCSV, dataset.fromCSV

system.dataset.getColumnHeaders

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

Description

Takes in a dataset and returns the headers as a python list.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.getColumnHeaders(dataset)

- Parameters

  Dataset dataset - The input dataset.

- Returns

  List - A list of column header strings.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example fetches the dataset from a Vision table, and prints the table headers as a
list.
# Fetch data from table component
data = event.source.parent.getComponent('Table').data
# Print dataset headers
print system.dataset.getColumnHeaders(data)
```

Keywords

system dataset getColumnHeaders, dataset.getColumnHeaders

system.dataset.setValue

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

Description

Takes a dataset and returns a new dataset with one value altered.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.setValue(dataset, rowIndex, columnName, value)

- Parameters

  Dataset dataset - The starting dataset. Will not be modified (datasets are immutable), but acts as the basis for the returned dataset.

  int rowIndex - The index of the row to set the value at (starting at 0)

  String columnName - The name of the column to set the value at. Case insensitive.

  PyObject value - The new value for the specified row/column.

- Returns

  Dataset - A new dataset, with the new value set at the given location.

- Scope

  Gateway, Vision Client, Perspective Session

Syntax

system.dataset.setValue(dataset, rowIndex, columnIndex, value)

- Parameters

  Dataset dataset - The starting dataset. Will not be modified (datasets are immutable), but acts as the basis for the returned dataset.

  Integer rowIndex - The index of the row to set the value at (starting at 0).

  Integer columnIndex - The index of the column to set the value at (starting at 0)

  Any value - The new value for the specified row/column.

- Returns

  Dataset - A new dataset, with the new value set at the given location.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet could be used for a Vision Button's actionPerformed event to change the
selected cell's value in a Table component to zero.
# Fetch table reference
table = event.source.parent.getComponent("Table")
# Fetch selected row and column
selRow = table.getSelectedRow()
selCol = table.getSelectedColumn()
# If row and column have been selected, update value in table to 0.
if selRow != -1 and selCol != -1:
   newData = system.dataset.setValue(table.data, selRow, selCol, 0.0)
   table.data = newData
```

Keywords

system dataset setValue, dataset.setValue

system.dataset.sort

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

Description

Takes a dataset and returns a sorted version of dataset. The sort order is determined by a single column. This works on numeric, as well as alphanumeric columns. When sorting alphanumerically, contiguous numbers are treated as a single number: you may recognize this as a "natural sort".

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Alphanumeric Sort

The table below represents an example of how alphanumeric values are sorted by the function. Where **Raw Column Values** represents an initial set of values, and the Sorted columns show how the function sorts in **Ascending** and **Descending** order.

Raw Column Values	Sorted - Ascending	Sorted - Descending
a1	a1	Z3
a22	A1	z3
Z3	a4	a77z99
z3	a7z9	a77z4
a4	a22	a22
a77z4	a77z4	a7z9
a77z99	a77z99	a4
a7z9	Z3	a1
A1	z3	A1

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.sort(dataset, keyColumn [, ascending, naturalOrdering])

- Parameters

  Dataset dataset - The dataset to sort.

  Integer | String keyColumn - The index of the column to sort on.

  Boolean ascending - True for ascending order, False for descending order. If omitted, ascending order will be used. [optional]

  The following feature is new in Ignition version **8.0.16**
  Click here to check out the other new features

  Boolean naturalOrdering - True for natural ordering, False for alphabetical ordering. Ignored if the sort column is a directly sortable data type. If omitted, defaults to True (natural ordering). [optional]

- Returns

  Dataset - A new sorted dataset.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code will take the data in a Vision Table component, sort it based on the column with
index 1,
# and then reinsert the sorted data into the same Table.

data = event.source.parent.getComponent('Table').data
newData = system.dataset.sort(data, 1)
event.source.parent.getComponent('Table').data = newData
```

Code Snippet

```
# This code will create a dataset in scripting, and then sort it based on the name of one of
the columns.
# It then inserts the sorted dataset into a table component.

# Initialize column headers and empty data list
headers = ["City", "Population", "Timezone", "GMTOffset"]
data = []
# Add rows, one by one, into data list
data.append(["New York", 8363710, "EST", -5])
data.append(["Los Angeles", 3833995, "PST", -8])
data.append(["Chicago", 2853114, "CST", -6])
data.append(["Houston", 2242193, "CST", -6])
data.append(["Phoenix", 1567924, "MST", -7])
# Convert headers and data lists into dataset
cities = system.dataset.toDataSet(headers, data)
# Sort the resulting dataset by city name
newData = system.dataset.sort(cities, "City")
# Write final dataset to a table
event.source.parent.getComponent('Table').data = newData
```

Keywords

system dataset sort, dataset.sort

system.dataset.toCSV

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

Description

Formats the contents of a dataset as CSV (comma separated values), returning the resulting CSV as a string. If the "forExport" flag is set, then the format will be appropriate for parsing using the system.dataset.fromCSV function.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.toCSV(dataset, showHeaders, forExport, localized)

- Parameters

  Dataset dataset - The dataset to export to CSV.

  Boolean showHeaders - If set to true(1), a header row will be present in the CSV. Default is true(1).

  Boolean forExport - If set to true(1), extra header information will be present in the CSV data which is necessary for the CSV to be compatible with the fromCSV method. Overrides showHeaders. Default is false(0).

  Boolean localized - If set to true(1), the string representations of the values in the CSV data will be localized. Defaul t is false(0).

- Returns

  String - The CSV data as a string

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet would run a SQL query against a database, and turn the results into a CSV
string. It would then store the resulting CSV to a file on the local hard drive.

results = system.db.runNamedQuery("Fetch Records",{})
csv = system.dataset.toCSV(dataset = results, showHeaders = True, forExport = False)
filePath = "C:\\output\\results.csv"
system.file.writeFile(filePath, csv)
```

Keywords

system dataset toCSV, dataset.toCSV

system.dataset.toDataSet

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

Description

This function is used to 1) convert PyDataSets to DataSets, and 2) create new datasets from raw Python lists. When creating a new dataset, headers should have unique names.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.toDataSet(dataset)

- Parameters

  PyDataSet dataset - A PyDataSet object to convert.

- Returns

  Dataset - The newly created dataset.

- Scope

  Gateway, Vision Client, Perspective Session

Syntax

system.dataset.toDataSet(headers, data)

- Parameters

  PySequence headers - The column names for the dataset to create.

  PySequence data - A list of rows for the new dataset. Each row must have the same length as the headers list, and each value in a column must be the same type.

- Returns

  Dataset - The newly created dataset.

- Scope

  All

Code Examples

Code Snippet

```
# This example create a single column dataset.
header = ['myColumn']
rows = [[1], [2]]
dataset = system.dataset.toDataSet(header, rows)
```

Code Snippet

```
# This second example shows how this function can be used to convert from a PyDataSet (which
is what system.db.runQuery returns) to a normal DataSet, which is the datatype of a Table
component's data property.

pyDataSet = system.db.runQuery("SELECT * FROM example1 LIMIT 100")
table = event.source.parent.getComponent("Table")
normalDataSet = system.dataset.toDataSet(pyDataSet)
table.data = normalDataSet
```

Keywords
system dataset toDataSet, dataset.toDataSet

system.dataset.toExcel

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

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

Description

Formats the contents of one or more datasets as an excel spreadsheet, returning the results as a string. Each dataset specified will be added as a worksheet in the Excel workbook.

This function replaces the deprecated system.dataset.dataSetToExcel function.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.toExcel(showHeaders, dataset, [nullsEmpty], [sheetNames])

- Parameters

  Boolean showHeaders - If True, the spreadsheet will include a header row. If False, the header row will be omitted.

  Object[] dataset - A sequence of one or more datasets, one for each sheet in the resulting workbook.

  Boolean nullsEmpty - If True, the spreadsheet will leave cells with NULL values empty, instead of allowing Excel to provide a default value like 0. Defaults to False. [Optional]

  The following feature is new in Ignition version **8.0.16**
  Click here to check out the other new features

  List sheetNames - Expects a list of strings, where each string is a name for one of the datasets. When used, there must be an equal number of string names in `sheetName` as there are datasets in the `dataset` parameter. Names provided in this parameter may be sanitized into acceptable Excel sheet names. [Optional]

- Returns

  Array - A byte array representing an Excel workbook.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet would run a SQL query against a database, and turn the results into a string
that is XML that Excel can open. It then writes the string to a file on the local hard drive.

results = system.db.runNamedQuery("Fetch Records",{})
spreadsheet = system.dataset.toExcel(True, [results])
filePath = "C:\\output\\results.xls"
system.file.writeFile(filePath, spreadsheet)
```

Keywords

system dataset toExcel, dataset.toExcel

system.dataset.toPyDataSet

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

Description

This function converts from a normal DataSet to a PyDataSet, which is a wrapper class which makes working with datasets more Python-esque. For more information on Datasets and PyDatasets, see the Datasets page.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.toPyDataSet(dataset)

- Parameters

  Dataset dataset - A DataSet object to convert into a PyDataSet.

- Returns

  PyDataSet - The newly created PyDataSet.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example script would be added to a button that is in the same container as the table
you are working with.
# It grabs the data of the Table component, and adds up the values in the column named
"Value", displaying the result to the user.

# Get a Table component's data
table = event.source.parent.getComponent("Table")
data = system.dataset.toPyDataSet(table.data)

# Loop through the data, summing the Value column
value = 0.0
for row in data:
   value += row["Value"]

# Show the user the sum of the Value column
system.gui.messageBox("The value is: %f" % value)
```

Keywords

system dataset toPyDataSet, dataset.toPyDataSet

system.dataset.updateRow

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

Description

Takes a dataset and returns a new dataset with a one row altered. To alter the row, this function takes a Python dictionary to represent the changes to make to the specified row. The keys in the dictionary are used to find the columns to alter.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See Altering a Dataset.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.dataset.updateRow(dataset, rowIndex, changes)

- Parameters

  Dataset dataset - The starting dataset. Will not be modified (datasets are immutable), but acts as the basis for the returned dataset.

  Integer rowIndex - The index of the row to update (starting at 0)

  PyDictionary changes - A Dictionary of changes to make. They keys in the dictionary should match column names in the dataset, and their values will be used to update the row.

- Returns

  Dataset - A new dataset with the values at the specified row updated according to the values in the dictionary.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example could be used to dynamically change the data that an Easy Chart displays.
# In this simple example, we assume that the chart is always configured to display a single
tank's level.
# This script would update the pen being displayed using a dynamic tank number.

# Generate new tag name and tag path
tankNumber = 5
newName = "Tank %d Level" % tankNumber
newPath = "Tanks/Tank %d/Level" % tankNumber

# Consolidate changes into a dictionary
updates = {"NAME": newName, "TAG_PATH":newPath}

# Update the Easy Chart
chart = event.source.parent.getComponent("Easy Chart")
newPens = system.dataset.updateRow(chart.tagPens, 0, updates)
chart.tagPens = newPens
```

Keywords

system dataset updateRow, dataset.updateRow

system.date

Date Functions

The following functions give you access to test and modify dates.

In This Section ...

system.date.*Between

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

Description

This function is a set of functions that include:

Function	Description
system.date.millisBetween	Calculates the number of whole milliseconds between two dates.
system.date.secondsBetween	Calculates the number of whole seconds between two dates.
system.date.minutesBetween	Calculates the number of whole minutes between two dates.
system.date.hoursBetween	Calculates the number of whole hours between two dates.
system.date.daysBetween	Calculates the number of whole days between two dates. Daylight savings changes are taken into account.
system.date.weeksBetween	Calculates the number of whole weeks between two dates.
system.date.monthsBetween	Calculates the number of whole months between two dates. Daylight savings changes are taken into account.
system.date.yearsBetween	Calculates the number of whole years between two dates. Daylight savings changes are taken into account.

Order does matter for the two dates passed in that we are calculating how much time has passed from date 1 to date 2. So, if date 2 is further in time than date 1, then a positive amount of time has passed. If date 2 is backwards in time from date 1, then a negative amount of time has passed.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.*between(date_1, date_2)

- Parameters

  Date date_1 - The first date to use.

  Date date_2 - The second date to use.

- Returns

  Int - An integer that is representative of the difference between two dates.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would grab the current time, and add 119 minutes to it, then calculate the number
# of hours between the two dates.

first = system.date.now()
second = system.date.addMinutes(first, 119)
print system.date.hoursBetween(first, second) # This would print 1 since it is only 1 whole hour.
```

Code Snippet

```
# This example would create two date objects, one on the 28th of May,
# and one on the 22nd of April, both in 2020. Because the second date is
# before the first date, a negative number will be returned.

first = system.date.getDate(2020, 4, 28)
second = system.date.getDate(2020, 3, 22)
print system.date.daysBetween(first, second) # This will print -36
```

Code Snippet

```
# This example can be placed on the action performed event of a button.
# It will then grab the week difference of two calendar components,
# and enter the value returned into a numeric text field.

first = event.source.parent.getComponent('Start Date Calendar').date
second = event.source.parent.getComponent('End Date Calendar').date
event.source.parent.getComponent('Numeric Text Field').intValue = system.date.weeksBetween
(first, second)
```

Keywords

system date *Between, date.*Between, date millisBetween, date.millisBetween, date secondsBetween, date.secondsBetween, date.minutesBetween, system.date.minutesBetween, date hoursBetween, date.hoursBetween, date weeksBetween, date.
weeksBetween, date monthsBetween, date.monthsBetween, date yearsBetween, date.yearsBetween

system.date.add*

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

Description

This function is a set of functions to add and subtract time that include:

Function	Description
system.date.addMillis	Add or subtract an amount of milliseconds to a given date and time.
system.date.addSeconds	Add or subtract an amount of seconds to a given date and time.
system.date.addMinutes	Add or subtract an amount of minutes to a given date and time.
system.date.addHours	Add or subtract an amount of hours to a given date and time.
system.date.addDays	Add or subtract an amount of days to a given date and time.
system.date.addWeeks	Add or subtract an amount of weeks to a given date and time.
system.date.addMonths	Add or subtract an amount of months to a given date and time. This function is unique since each month can have a variable number of days. For example, if the date passed in is March 31st, and we add one month, April does not have a 31st day, so the returned date will be the proper number of months rounded down to the closest available day, in this case April 30th.
system.date.addYears	Add or subtract an amount of years to a given date and time.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.add*(date, value)

- Parameters

  Date date- The starting date.

  Int value - The number of units to add, or subtract if the value is negative.

- Returns

  Date - A new date object offset by the integer passed to the function.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would add two days to the date passed in.

today = system.date.now()
twoDaysFromToday = system.date.addDays(today, 2)
```

Code Snippet

```
# This example would subtract twenty minutes from a date object we create.
# Even though our original date starts on the 28th, it starts at midnight,
# so subtracting 20 minutes puts it at the previous day.

date = system.date.getDate(2018, 4, 28) #This would print out like Mon May 28 00:00:00 PDT 2018
print system.date.addMinutes(date, -20) #This will print Sun May 27 23:40:00 PDT 2018
```

Code Snippet

```
# This example can be placed on the property change script of one calendar component.
# It will then automatically set a second calendar component two weeks in advanced of the first
calendars selected date.
if event.propertyName == "date":
        date = event.newValue
        event.source.parent.getComponent('End Date Calendar').date = system.date.addWeeks(date,
2)
```

Keywords

system date add*, date.add*, date addMillis, date.addMillis, date addSeconds, date.addSeconds, date addMinutes, date.addMinutes, date addHours, date.addHours, date addDays, date.addDays, date addWeeks, date.addWeeks, date addMonths, date.addMonths, date addYears, date.addYears

system.date.format

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

Description

Returns the given 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.

The placeholders are:

Symbol	Description	Presentation	Examples	Other Notes
`G`	Era designator	Text	`G=AD`	
`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)
`M`	Month in year	Month	`MMMM=July`; `MMM=Jul`; `MM=07`	
`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`	
`D`	Day in year	Number	`189`	
`d`	Day in month	Number	`10`	
`F`	Day of week in month	Number	`2`	2nd Sunday of the month
`E`	Day name in week	Text	`EEEE=Tuesday`; `E=Tue`	
`u`	Day number of week	Number	`1`	(1 = Monday, ..., 7 = Sunday)
`a`	Am/Pm marker	Text	`PM`	
`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`	Minute in hour	Number	`30`	
`s`	Second in minute	Number	`55`	
`S`	Millisecond	Number	`978`	
`z`	Time zone	General time zone	`zzzz=Pacific Standard Time`; `z=PST`	
`Z`	Time zone	RFC 822 time zone	`Z=-0800`	
`X`	Time zone	ISO 8601 time zone	`X=-08`; `XX=-0800`; `XXX=-08:00`	

Expert Tip: This function uses the Java class java.text.SimpleDateFormat internally, and will accept any valid format string for that class.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.format(date, format)

- Parameters

  Date date - The date to format.

  String format - A format string such as "yyyy-MM-dd HH:mm:ss".

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

As of version 8.0.8 the format argument is optional. The default is "yyyy-MM-dd HH:mm:ss".

- Returns

  String - A string representing the formatted datetime

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would format the current date to look like "01/01/01"

today = system.date.now()
print system.date.format(today, "yy/MM/dd")
#This printed 16/04/01
```

Code Snippet

```
# This example would format the current date to look like "2001-01-31 16:59:59"
# This is a standard format that all databases recognize.

today = system.date.now()
print system.date.format(today, "yyyy-MM-dd HH:mm:ss")
```

Keywords

system date format, date.format

system.date.fromMillis

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

Description

Creates a date object given a millisecond value

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.fromMillis(millis)

- Parameters

  Long millis- The number of milliseconds elapsed since January 1, 1970, 00:00:00 UTC (GMT)

- Returns

  Date - A new date object

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print out the date "Fri Aug 18 14:35:25 PDT 2017"

print system.date.fromMillis(1503092125000)
```

Keywords

system date fromMillis, date.fromMillis

system.date.get*

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

Description

This function is a set of functions that include:

Function	Description
system.date.getMillis	Extracts the milliseconds from a date, ranging from 0-999.
system.date.getSecond	Extracts the second from a date, ranging from 0-59.
system.date.getMinute	Extracts the minutes from a date, ranging from 0-59.
system.date.getHour12	Extracts the hour from a date. Uses a 12 hour clock, so noon and midnight are returned as 0.
system.date.getHour24	Extracts the hour from a date. Uses a 24 hour clock, so midnight is zero.
system.date.getDayOfWeek	Extracts the day of the week from a date. Sunday is day 1, Saturday is day 7.
system.date.getDayOfMonth	Extracts the day of the month from a date. The first day of the month is day 1.
system.date.getDayOfYear	Extracts the day of the year from a date. The first day of the year is day 1.
system.date.getMonth	Extracts the month from a date, where January is month 0.
system.date.getQuarter	Extracts the quarter from a date, ranging from 1-4.
system.date.getYear	Extracts the year from a date.
system.date.getAMorPM	Returns a 0 if the time is before noon, and a 1 if the time is after noon.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.get* (date)

- Parameters

  Date date - The date to use.

- Returns

  Int - An integer that is representative of the extracted value.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would grab the current time, and print the current month.

date = system.date.now()
print system.date.getMonth(date) #This would print the current month.
```

Code Snippet

```
# This example would create a date object, and print out the quarter of that date.

date = system.date.getDate(2017, 3, 15) # This would print "Mon April 15 00:00:00 PDT 2016"
print system.date.getQuarter(date) # This will print 2
```

Code Snippet

```
# This example can be placed on the action performed event of a button.
# It will then grab the day of the week of the calendar component,
# and enter the value returned into a numeric text field.

date = event.source.parent.getComponent('Calendar').date
event.source.parent.getComponent('Numeric Text Field').intValue = system.date.getDayOfWeek(date)
```

Keywords

system date get*, date.get*, date getMillis, date.getMillis, date getSecond, date.getSecond, date getMinute, date.getMinute, date getHour12, date.getHour12, date getHour24, date.getHour24,date getDayOfWeek, date.getDayOfWeek, date getDayOfMonth, date.getDayOfMonth, date getDayOfYear, date.getDayOfYear, date getMonth, date.getMonth, date getQuarter, date.getQuarter, date getYear, date.getYear, date getAMorPM, date.getAMorPM

system.date.getDate

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

Description

Creates a new Date object given a year, month and a day. The time will be set to midnight of that day.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.getDate(year, month, day)

- Parameters

  Int year - The year for the new date.

  Int month - The month of the new date. January is month 0.

  Int day - The day of the month for the new date. The first day of the month is day 1.

- Returns

  Date - A new date, set to midnight of that day.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will create a new date object set to January 1st, 2017.

date = system.date.getDate(2017, 0, 1)
print date
```

Keywords

system date getDate, date.getDate

system.date.getTimezone

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

Description

Returns the ID of the current timezone.

*This list is subject to change depending on the exact version of java that is installed.

Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca

```
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
```

```
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Kralendijk
America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Lower_Princes
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
```

```
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok
Arctic/Longyearbyen
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Chita
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Katmandu
Asia/Khandyga
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
```

```
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Srednekolymsk
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Ust-Nera
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
Brazil/DeNoronha
```

```
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/East-Saskatchewan
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba
EET
EST5EDT
Egypt
Eire
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+10
Etc/GMT+11
Etc/GMT+12
Etc/GMT+2
Etc/GMT+3
Etc/GMT+4
Etc/GMT+5
Etc/GMT+6
Etc/GMT+7
Etc/GMT+8
Etc/GMT+9
Etc/GMT-0
Etc/GMT-1
Etc/GMT-10
Etc/GMT-11
Etc/GMT-12
Etc/GMT-13
Etc/GMT-14
Etc/GMT-2
Etc/GMT-3
Etc/GMT-4
Etc/GMT-5
Etc/GMT-6
Etc/GMT-7
Etc/GMT-8
Etc/GMT-9
Etc/GMT0
Etc/Greenwich
Etc/UCT
Etc/UTC
Etc/Universal
Etc/Zulu
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Busingen
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
```

```
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Simferopol
Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB
GB-Eire
GMT
GMT0
Greenwich
Hongkong
Iceland
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
```

```
Pacific/Bougainville
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
Pacific/Yap
Poland
Portugal
ROK
Singapore
SystemV/AST4
SystemV/AST4ADT
SystemV/CST6
SystemV/CST6CDT
SystemV/EST5
SystemV/EST5EDT
SystemV/HST10
SystemV/MST7
SystemV/MST7MDT
SystemV/PST8
SystemV/PST8PDT
SystemV/YST9
SystemV/YST9YDT
Turkey
UCT
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Pacific-New
US/Samoa
UTC
Universal
```

W-SU
WET
Zulu
EST
HST
MST
ACT
AET
AGT
ART
AST
BET
BST
CAT
CNT
CST
CTT
EAT
ECT
IET
IST
JST
MIT
NET
NST
PLT
PNT
PRT
PST
SST
VST

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.getTimezone( )

- Parameters

  none

- Returns

  String - A representation of the current timezone.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print out your current Timezone ID.
# If your Client and Gateway are in different timezones, the returned value will be
# dependent on where this script is run.
# IE: in a button on a client will return the client timezone. On a Gateway script will
# return the Gateway timezone.

print system.date.getTimezone()
```

Keywords

system date getTimezone, date.getTimezone

system.date.getTimezoneOffset

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

Description

Returns the current timezone's offset versus UTC for a given instant, taking Daylight Savings Time into account.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.getTimezoneOffset([date])

- Parameters

  Date date- The instant in time for which to calculate the offset. Uses now() if omitted. [optional]

- Returns

  Double - The timezone offset compared to UTC, in hours.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will give the timezone offset using the date February 22, 2017
# and the computers current timezone.

date = system.date.getDate(2017, 1, 22)
print system.date.getTimezoneOffset(date) # returns -8.0 if you are in Pacific Daylight Time
```

Keywords

system date getTimezoneOffset, date.getTimezoneOffset

system.date.getTimezoneRawOffset

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

Description

Returns the current timezone offset versus UTC, not taking daylight savings into account.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.getTimezoneRawOffset()

- Parameters

  none

- Returns

  Double - The timezone offset.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will give the Raw timezone offset (ignoring daylight savings) for the computers
current timezone.

print system.date.getTimezoneRawOffset() # returns -8.0 (if you are in the Pacific Timezone)
```

Keywords

system date getTimezoneRawOffset, date.getTimezoneRawOffset

system.date.isAfter

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

Description

Compares to dates to see if date_1 is after date_2.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.isAfter(date_1, date_2)

- Parameters

  Date date_1 - The first date.

  Date date_2 - The second date.

- Returns

  Bool - True (1) if date_1 is after date_2, false (0) otherwise.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This will compare if the first date is after the second date, which it is.

first = system.date.getDate(2018, 4, 28)
second = system.date.getDate(2018, 3, 22)
print system.date.isAfter(first, second) #Will print true.
```

Keywords

system date isAfter, date.isAfter

system.date.isBefore

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

Description

Compares to dates to see if date_1 is before date_2.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.isBefore(date_1, date_2)

- Parameters

  Date date_1 - The first date.

  Date date_2 - The second date.

- Returns

  Bool - True (1) if date_1 is before date_2, false (0) otherwise.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This will compare if the first date is before the second date, which it is not.

first = system.date.getDate(2018, 4, 28)
second = system.date.getDate(2018, 3, 22)
print system.date.isBefore(first, second) #Will print false.
```

Keywords

system date isBefore, date.isBefore

system.date.isBetween

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

Description

Compares two dates to see if a target date is between two other dates.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.isBetween(target_date, start_date, end_date)

- Parameters

  Date target_date - The date to compare.

  Date start_date - The start of a date range.

  Date end_date - The end of a date range. This date must be after the start date.

- Returns

  Bool - True (1) if target_date is >= start_date and target_date <= end_date, false (0) otherwise.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This will compare if the first date is between the other dates, which it is not.
# Note that if the end date is before the start date, this function will always return False

target = system.date.getDate(2017, 4, 28)
start = system.date.getDate(2017, 3, 22)
end = system.date.getDate(2017, 4, 22)
print system.date.isBetween(target, start, end) #Will print false.
```

Keywords

system date isBetween, date.isBetween

system.date.isDaylightTime

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

Description

Checks to see if the current timezone is using daylight savings time during the date specified.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.isDaylightTime([date])

- Parameters

  Date date - The date you want to check if the current timezone is observing daylight savings time. Uses now() if omitted. [optional]

- Returns

  Bool - True (1) if date is observing dalight savings time in the current timezone, false (0) otherwise.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
date = system.date.getDate(2018, 6, 28)
print system.date.isDaylightTime(date) #Will print True in the US Pacific Timezone.
```

Keywords

system date isDaylightTime, date.isDaylightTime

system.date.midnight

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

Description

Returns a copy of a date with the hour, minute, second, and millisecond fields set to zero.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.midnight(date)

- Parameters

  Date date- The starting date.

- Returns

  Date - A new date, set to midnight of the day provided

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print out the current date with the time set to midnight.

date = system.date.now()
print system.date.midnight(date)
```

Keywords

system date midnight, date.midnight

system.date.now

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

Description

Returns a java.util.Date object that represents the current time according to the local system clock.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.now()

- Parameters

  none

- Returns

  Date - A new date, set to the current date and time.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will set a calendar component to the current date and time.

event.source.parent.getComponent('Calendar').date = system.date.now()
```

Keywords

system date now, date.now

system.date.parse

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

Description

Attempts to parse a string and create a Date. Causes ParseException if the date dateString parameter is in an unrecognized format.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.parse(dateString, formatString, locale)

- Parameters

  String dateString - The string to parse into a date.

  String formatString - Format string used by the parser. Default is "yyyy-MM-dd HH:mm:ss". Click here for a list of formatString placeholder characters. [optional]

  Object locale - Locale used for parsing. Can be the locale name such as 'fr', or the Java Locale such as 'Locale.French'. Default is 'Locale.English'. Click here for a list of supported locales and and values. [optional]

- Returns

  Date - The parsed date.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will return a date object set to the given date and time, May 28th, 1992 at 4:
22am.
print system.date.parse('May 28, 1992 4:22', 'MMMM dd, yyyy hh:mm')
```

Code Example - Using the Locale Parameter

```
# This example demonstrates the locale parameter to parse a French date
print system.date.parse("juillet 15, 2015 10:32:15", "MMMM dd, yyyy hh:mm:ss", "fr")

# If using the Java Locale, then you must import from the Locale class. The following example
parses a German date.
from java.util import Locale
print system.date.parse('21-Februar-2017 04:22:00', 'dd-MMMM-yyyy HH:mm:ss', Locale.GERMAN)
```

Keywords

system date parse, date.parse

system.date.setTime

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

Description

Takes in a date, and returns a copy of it with the time fields set as specified.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.setTime(date, hour, minute, second)

- Parameters

  Date date - The starting date.

  Int hour - The hours (0-23) to set.

  Int minute - The minutes (0-59) to set.

  Int second - The seconds (0-59) to set.

- Returns

  Date - A new date, set to the appropriate time.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will set the date object to the current date with the time set to 01:37 in the
morning and 44 seconds.

date = system.date.getDate(2018, 6, 29) #getDate is zero based, so a month parameter of 6 will
return July.
print system.date.setTime(date, 1, 37, 44) #This will print Fri July 29 01:37:44 PDT 2018
```

Keywords

system date setTime, date.setTime

system.date.toMillis

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

Description

Converts a Date object to its millisecond value elapsed since January 1, 1970, 00:00:00 UTC (GMT)

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.date.toMillis(date)

- Parameters

  Date date - The date object to convert.

- Returns

  Long - an 8-byte integer representing the number of millisecond elapsed since January 1, 1970, 00:00:00 UTC (GMT)

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will take the date Fri Aug 18 14:35:25 PDT 2017,
# and print out 1503092125000

date = system.date.getDate(2017, 7, 18)
datetime = system.date.setTime(date, 14, 35, 25)
print system.date.toMillis(datetime)
```

Keywords

system date toMillis, date.toMillis

system.db

Database Functions

The following functions give you access to view and modify data in the database.

In This Section ...

system.db.addDatasource

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

Description

Adds a new database connection in Ignition.

Client Permission Restrictions

Permission Type: Datasource 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.db.addDatasource(jdbcDriver, name, description, connectUrl, username, password, props, validationQuery, maxConnections)

- Parameters

  String jdbcDriver - The name of the JDBC driver configuration to use. Available options are based off the JDBC driver configurations on the the gateway. Required.

  String name - The datasource name. Required.

  String description - Description of the datasource

  String connectUrl - Default is the connect URL for JDBC driver.

  String username - Username to login to the datasource with.

  String password - Password for the login.

  String props - The extra connection parameters.

  String validationQuery - Default is the validation query for the JDBC driver.

  Integer maxConnections - Default is 8.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Adding a MySQL Database to a Gateway

```
system.db.addDatasource(jdbcDriver="myJDBCDriver", name="NewDatabase",
connectURL="jdbc:mysql://localhost:3306/test", username="root",
password="password", props="zeroDateTimeBehavior=convertToNull;")
```

Keywords

system db addDatasource, db.addDatasource

system.db.beginNamedQueryTransaction

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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

Description

Begins a new database transaction using Named Queries. Database transactions are used to execute multiple queries in an atomic fashion. After executing queries, you must either commit the transaction to have your changes take effect, or rollback th e transaction which will make all operations since the last commit not take place. The transaction is given a new unique string code, which is then returned. You can then use this code as the tx argument for other system.db.* function calls to execute various types of queries using this transaction.

An open transaction consumes one database connection until it is closed. Because leaving connections open indefinitely would exhaust the connection pool, each transaction is given a timeout. Each time the transaction is used, the timeout timer is reset. For example, if you make a transaction with a timeout of one minute, you must complete that transaction within a minute. If a transaction is detected to have timed out, it will be automatically closed and its transaction id will no longer be valid.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax - Vision

system.db.beginNamedQueryTransaction([database], [isolationLevel], [timeout])

- Parameters

  String database - The name of the database connection to create a transaction in. If omitted, uses the project's default connection.

  Integer/Constant isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db. READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db. SERIALIZABLE. If omitted, uses system.db.READ_COMMITTED

  Long timeout timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back. If omitted, defaults to 30,000.

- Returns

  String - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

  Vision Client

Syntax - Perspective and Gateway

system.db.beginNamedQueryTransaction(project, database, [isolationLevel], [timeout])

- Parameters

  String project - The name of the project that contains the named query.

  String database - The name of the database connection to create a transaction in.

  Integer/Constant isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db.READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db.SERIALIZABLE. If omitted, uses system.db.READ_COMMITTED

  Long timeout timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back. If omitted, defaults to 30,000.

- Returns

  String - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

  Gateway, Perspective Session

Isolation Level Values

The following table lists each value of the isolationLevel parameter and its associated level. Either the integer value or constant may be passed. Note that some JDBC drivers only support some levels, so the driver's documentation should be consulted. Isolation levels are well documented online, but the following link is a great starting point: Data Concurrency and Consistency

Isolation Level	Int Value	Constant
Read Uncommitted	1	system.db.READ_UNCOMMITTED
Read Committed	2	system.db.READ_COMMITTED
Repeatable Read	4	system.db.REPEATABLE_READ
Serializable	8	system.db.SERIALIZABLE

Code Examples

Code Snippet - Running Named Query Using Named Query Transactions

```
# This example would start a transaction and check a screen to see if the transaction should
be completed or reversed (rolled back).
# The example assumes you have several components on screen and a Named query that takes in
an ID and a string value.

# Get details from the screen: Numeric Text Field, Text Field, Checkbox
idEntry = event.source.parent.getComponent('ID Field').intValue
valueEntry = event.source.parent.getComponent('Value Field').text
shouldRollback = event.source.parent.getComponent('CheckBox').selected

# Begin the transaction
datasource = "MYSQL"
isolationLevel = system.db.READ_COMMITTED
timeout = 60000
txNumber = system.db.beginNamedQueryTransaction(datasource, isolationLevel, timeout)

# start by running a Named Query against the transaction
namedQueryPath = "InsertQueries/AddValues"
params = {"id":idEntry, "value":valueEntry}
system.db.runNamedQuery(namedQueryPath, params, txNumber)

# check the window to see if the user selected to cancel the transaction
if shouldRollback:
        # cancel the transaction
        system.db.rollbackTransaction(txNumber)
        print "Transaction rolled back"
else:
        # complete the transaction
        system.db.commitTransaction(txNumber)
        print "Transaction committed"

# Close the transaction now that we are done
system.db.closeTransaction(txNumber)
```

Keywords

system db beginNamedQueryTransaction, db.beginNamedQueryTransaction

system.db.beginTransaction

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

Description

Begins a new database transaction for using run* and runPrep* queries. Database transactions are used to execute multiple queries in an atomic fashion. After executing queries, you must either commit the transaction to have your changes take effect, or rollback the transaction which will make all operations since the last commit not take place. The transaction is given a new unique string code, which is then returned. You can then use this code as the tx argument for other system.db.* function calls to execute various types of queries using this transaction.

An open transaction consumes one database connection until it is closed. Because leaving connections open indefinitely would exhaust the connection pool, each transaction is given a timeout. Each time the transaction is used, the timeout timer is reset. For example, if you make a transaction with a timeout of one minute, you must use that transaction at least once a minute. If a transaction is detected to have timed out, it will be automatically closed and its transaction id will no longer be valid.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.beginTransaction(database, isolationLevel, timeout)

- Parameters

  String database - The name of the database connection to create a transaction in.

  Integer/Constant isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db.READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db.SERIALIZABLE

  Long timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back.

- Returns

  String - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

  Gateway

Syntax

system.db.beginTransaction(database, isolationLevel, timeout)

- Parameters

  String database - The name of the database connection to create a transaction in. Use "" for the project's default connection.

  Integer/Constant isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db.READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db.SERIALIZABLE

  Long timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back.

- Returns

  String - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

  Vision Client, Perspective Session

Isolation Level Values

The following table lists each value of the isolationLevel parameter and its associated level. Either the integer value or constant may be passed. Note that some JDBC drivers only support some levels, so the driver's documentation should be consulted. Isolation levels are well documented online, but the following link is a great starting point: Data Concurrency and Consistency

Isolation Level	Int Value	Constant
Read Uncommitted	1	system.db.READ_UNCOMMITTED
Read Committed	2	system.db.READ_COMMITTED
Repeatable Read	4	system.db.REPEATABLE_READ
Serializable	8	system.db.SERIALIZABLE

Code Examples

Code Snippet - Running a Query Using Query Transactions

```
# This example would start a transaction with a 5 second timeout against the project's
default database, using the default isolation level. Then it executes a series of update
calls, and commits and closes the transaction.

txId = system.db.beginTransaction(timeout=5000)
status=2

for machineId in range(8):
   system.db.runPrepUpdate("UPDATE MachineStatus SET status=? WHERE ID=?",
      args=[status, machineId], tx=txId)

system.db.commitTransaction(txId)
system.db.closeTransaction(txId)
```

Keywords

system db beginTransaction, db.beginTransaction

system.db.clearAllNamedQueryCaches

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

Description

This clears the caches of all Named Queries in a project. If called from the Shared Scope (i.e., Tag Event Scripts, Alarm Pipelines, etc.) then the name of the project must be passed as a parameter.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax - Project Scope

system.db.clearAllNamedQueryCaches()

- Parameters

  none

- Returns

  No Return Value

- Scope

  Vision Clients

Syntax - Shared Scope

system.db.clearAllNamedQueryCaches(project)

- Parameters

  String project - The Project that contains the named query whose cache needs to be cleared.

- Returns

  No Return Value

- Scope

  Gateway, Perspective Session

Code Examples

Example - Clear All Named Query Cache for a Specific Project

```
# Calling this simply clears all Named Query Caches.
# This is assumed to run in the Shared Scope, so the name of the project must be included.
system.db.clearAllNamedQueryCaches("myProjectName")
```

Example - Clear All Named Query Cache

```
# If multiple Named Queries with varying parameters are called in a single script, then
clearAllNamedQueryCaches can be used to free up the memory used by all of the newly created
caches.
# This example is assumed to run in the Project Scope, so the project parameter may be
omitted.

# This creates one cache.
params = {"param1":"A"}
system.db.runNamedQuery("myUpdateQuery", params)

# This creates a separate cache.
params = {"param1":"B"}
system.db.runNamedQuery("anotherUpdateQuery", params)

# Clear all of the caches from the current project. Note that all caches are cleared,
including those generated from elsewhere on the Gateway.
system.db.clearAllNamedQueryCaches()
```

Keywords

system db clearAllNamedQueryCaches, db.clearAllNamedQueryCaches

system.db.clearNamedQueryCache

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

Description

This clears the cache of a Named Query. If called from the Shared Scope (i.e., Tag Event Scripts, Alarm Pipelines, etc.) then the name of the project must be passed as a parameter.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax - Project Scope

system.db.clearNamedQueryCache(path)

- Parameters

  String path - The Path to the named query we want to clear the cache of.

- Returns

  No Return Value

- Scope

  Vision Client, Perspective Session

Syntax - Shared Scope

system.db.clearNamedQueryCache(project, path)

- Parameters

  String project - The Project that contains the named query whose cache needs to be cleared.

  String path - The Path to the named query we want to clear the cache of.

- Returns

  No Return Value

- Scope

  Gateway

Code Examples

Example - Clear Named Query Cache for a Specific Project

```
# Calling this simply clears all Named Query Caches.
# This example is being called from the Shared Scope. If called from the Project Scope, the
projectName parameter should be omitted.


projectName = "myProject"
namedQueryPath = "folder/selectFromInventory"

system.db.clearNamedQueryCache(projectName, namedQueryPath)
```

Example - Clear Named Query Cache

```
# If the same Named Queried is called multiple times with different parameters in a single
script, then we can clear the caches once we're done with the following.
# This example assumes the script is running in the Project Scope. If called from the Shared
Scope, the name of the project would need to be included.

namedQueryPath = "myUpdateQuery"

# This creates one cache.
params = {"param1":"A"}
system.db.runNamedQuery(namedQueryPath, params)

# This creates a separate cache.
params = {"param1":"B"}
system.db.runNamedQuery(namedQueryPath, params)

# Clear all of the caches from the specified Named Query. Note that all caches are cleared,
including those generated from elsewhere on the Gateway.
system.db.clearNamedQueryCache(namedQueryPath)
```

Keywords

system db clearNamedQueryCache, db.clearNamedQueryCache

system.db.closeTransaction

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

Description
Closes the transaction with the given ID. Note that you must commit or rollback the transaction before you close it. Closing the transaction will return its database connection to the pool. The transaction ID will no longer be valid.

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

Syntax

system.db.closeTransaction(tx)

- Parameters

  String tx - The transaction ID.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no example available for this function.

Keywords
system db closeTransaction, db.closeTransaction

system.db.commitTransaction

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

Description

Performs a commit for the given transaction. This will make all statements executed against the transaction since its beginning or since the last commit or rollback take effect in the database. Until you commit a transaction, any changes that the transaction makes will not be visible to other connections. Note that if you are done with the transaction, you must close it after you commit it.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.db.commitTransaction(tx)

- Parameters

  String tx - The transaction ID.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples available for this function.

Keywords

system db commitTransaction, db.commitTransaction

system.db.createSProcCall

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

Description

Creates an SProcCall object, which is a stored procedure call context. This is an object that is used to configure a call to a stored procedure. Once configured, you'd use system.db.execSProcCall to call the stored procedure. The call context object then holds any results from the stored procedure. The SProcCall object has the following functions used for registering parameters:

SPRocCall.registerInParam(index OR name, typeCode, value)

SPRocCall.registerOutParam(index OR name, typeCode)

SPRocCall.registerReturnParam(typeCode)

These functions are used to register any in/out parameters for the stored procedure. Parameters can be referenced by index (starting at 1, not 0), or by name. To register an in/out parameter, you simply register it twice - once as an input parameter with the value you'd like to pass to the stored procedure, and once as an output parameter. Note that not all JDBC drivers support named procedure parameters. If your function returns a value, you must use registerReturnParam to specify the datatype of the returned value. Also be aware that this is different from stored procedures that return a result set, which doesn't require any setup on the SProcCall object. Some database systems call stored procedures that return a value "functions" instead of "proced ures". For all of these functions, you'll need to specify a type code. These are codes defined by the JDBC specification. For your convenience, the codes exist as constants in the system.db namespace. Each type code will be mapped to a database-specific type by the JDBC driver. Not all type codes will be recognized by all JDBC drivers. The following type code constants are available for use in createSProcCall:

BIT	REAL	LONGVARCHAR	LONGVARBINARY
TINYINT	DOUBLE	DATE	NULL
SMALLINT	NUMERIC	TIME	ROWID
INTEGER	DECIMAL	TIMESTAMP	CLOB
BIGINT	CHAR	BINARY	NCLOB
FLOAT	VARCHAR	VARBINARY	BLOB
NCHAR	NVARCHAR	LONGNVARCHAR	BOOLEAN

The following type code constants are available for other uses, but are not supported by createSProcCall:

ORACLE_CURSOR	DISTINCT	STRUCT	REF
JAVA_OBJECT	SQLXML	ARRAY	DATALINK
OTHER			

Once the call context has been executed, you can retrieve the result set, return value, and output parameter values (if applicable) by calling the following functions:

SProcCall.getResultSet() - returns a dataset that is the resulting data of the stored procedure, if any.

SProcCall.getUpdateCount() - returns the number of rows modified by the stored procedure, or -1 if not applicable.

SProcCall.getReturnValue() - returns the return value, if registerReturnParam had been called.

SProcCall.getOutParamValue(index OR name) - returns the value of the previously registered out-parameter.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.db.createSProcCall(procedureName, database, tx, skipAudit)

- Parameters

  String procedureName - The named of the stored procedure to call.

  String database - The name of the database connection to execute against.

  String tx - A transaction identifier. If omitted, the call will be executed in its own transaction.

  boolean skipAudit - A flag which, if set to true, will cause the procedure call to skip the audit system. Useful for some queries that have fields which won't fit into the audit log.

- Returns

  SProcCall - A stored procedure call context, which can be configured and then used as the argument to system.db. execSProcCall.

- Scope

  Gateway

Syntax

system.db.createSProcCall(procedureName, database, tx, skipAudit)

- Parameters

  String procedureName - The named of the stored procedure to call.

  String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - A transaction identifier. If omitted, the call will be executed in its own transaction.

  boolean skipAudit - A flag which, if set to true, will cause the procedure call to skip the audit system. Useful for some queries that have fields which won't fit into the audit log.

- Returns

  SProcCall - A stored procedure call context, which can be configured and then used as the argument to system.db. execSProcCall.

- Scope

  Vision Client, Perspective Session

Code Examples

Code Snippet - Creating Stored Procedure Call

```
# This example would call a stored procedure named "start_batch" against the current
project's default database connection that had no input or output parameters, and did not
return any values or results:

call = system.db.createSProcCall("start_batch")
system.db.execSProcCall(call)
```

Code Snippet - Creating Stored Procedure Call

```
# This example would call a stored procedure "get_shift_workers" with no arguments, which
returned a result set of employees for the current shift. It then pushes the resulting
dataset into a Table component:

call = system.db.createSProcCall("get_shift_workers")
system.db.execSProcCall(call)

results = call.getResultSet()
table = event.source.parent.getComponent("Table")
table.data = results
```

Code Snippet - Creating Stored Procedure Call With Stored Procedure Parameters

```
# This example would call a stored procedure that took two arguments, the first an integer
and the second a string. It also is configured to return an integer value.

call = system.db.createSProcCall("perform_calculation")
call.registerReturnParam(system.db.INTEGER)
call.registerInParam(1, system.db.INTEGER, 42)
call.registerInParam(2, system.db.VARCHAR, "DC-MODE")

system.db.execSProcCall(call)

# Print the result to the console
print call.getReturnValue()
```

Code Snippet - Creating Stored Procedure Call With Stored Procedure Parameters

```
# This example would do the same as the one above, except for a stored procedure that
returned its value using an out-parameter. It also uses named argument names instead of
indexed arguments.

call = system.db.createSProcCall("perform_calculation")
call.registerInParam("arg_one", system.db.INTEGER, 42)
call.registerInParam("arg_two", system.db.VARCHAR, "DC-MODE")
call.registerOutParam("output_arg", system.db.INTEGER)

system.db.execSProcCall(call)

# Print the result to the console
print call.getOutParamValue("output_arg")
```

Keywords

system db createSProcCall, db.createSProcCall

system.db.dateFormat

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

Description

This function is used to format Dates nicely as strings. It uses a format string to guide its formatting behavior. Learn more about date formatting in Dates

Expert Tip: This function uses the Java class java.text.SimpleDateFormat internally, and will accept any valid format string for that class.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.db.dateFormat(date, formatPattern)

- Parameters

  Date date - The Date object that you'd like to format

  String formatPattern - A format pattern string to apply.

- Returns

  String - The date as a string formatted according to the format pattern.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will display a message box on a button press that displays the selected date
(without the time)
# from a Calendar component, in a format like "Feb 3, 2009"
date = event.source.parent.getComponent("Calendar").latchedDate
toDisplay = system.db.dateFormat(date, "MMM d, yyyy")
system.gui.messageBox("The date you selected is: %s" % toDisplay)
```

Code Snippet

```
# This example would do the same as the one above, but also display the time, in a format
like: "Feb 3, 2009 8:01pm"
date = event.source.parent.getComponent("Calendar").latchedDate
toDisplay = system.db.dateFormat(date, "MMM d, yyyy hh:mm a")
system.gui.messageBox("The date you selected is: %s" % toDisplay)
```

Code Snippet

```
# This example would take two dates from two Popup Calendar components, format them in a
manner that the database understands,
# and then use them in a SQL query to limit the results to a certain date range.
startDate = event.source.parent.getComponent("StartDate").date
endDate = event.source.parent.getComponent("EndDate").date
startDate = system.db.dateFormat(startDate, "yyyy-MM-dd HH:mm:ss")
endDate = system.db.dateFormat(endDate, "yyyy-MM-dd HH:mm:ss")
query = ("SELECT * FROM mytable WHERE t_stamp >= '%s' AND t_stamp <= '%s'" % (startDate,
endDate))
results = system.db.runQuery(query)
event.source.parent.getComponent("Table").data = results
```

Keywords

system db dateFormat, db.dateFormat

system.db.execSProcCall

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

Description

Executes a stored procedure call. The one parameter to this function is an SProcCall - a stored procedure call context. See the description of system.db.createSProcCall for more information and examples.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.db.execSProcCall(callContext)

- Parameters

  SProcCall callContext - A stored procedure call context, with any input, output, and/or return value parameters correctly configured. Use system.db.createSProcCall to create a call context.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples available for this function.

Keywords

system db execSProcCall, db.execSProcCall

system.db.getConnectionInfo

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

Description

Returns a dataset of information about a single database connection, as specified by the name argument.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.db.getConnectionInfo(name)

- Parameters

  String name - The name of the database connection to find information about.

- Returns

  Dataset - A dataset containing information about the named database connection, or an empty dataset if the connection wasn't found.

- Scope

  Gateway, Vision Client, Perspective Session

⚠ The database connection used when called from the Gateway scope is the connection configured on the Gateway scripting project

Code Examples

Code Snippet - Getting Database Connection Information

```
# This example checks the database connection type and selects a query format that matches.

connectionInfo = system.db.getConnectionInfo()
dbType = connectionInfo.getValueAt(0, "DBType")
if dbType == "MYSQL":
        # mysql format for a column with a space in the name
        query = "SELECT `amps value` FROM pumps"
else:
        # mssql format for a column with a space in the name
        query = "SELECT [amps value] FROM pumps"
```

Keywords

system db getConnectionInfo, db.getConnectionInfo

system.db.getConnections

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

Description
Returns a dataset of information about each configured database connection. Each row represents a single connection.

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

Syntax

system.db.getConnections()

- Parameters

  None

- Returns

  Dataset - A dataset, where each row represents a database connection.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no code examples available for this function.

Keywords
system db getConnections, db.getConnections

system.db.refresh

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

Description

This function will cause a Vision component binding to execute immediately. This is most often used for bindings that are set to Polling - Off. In this way, you cause a binding to execute on demand, when you know that the results of its query will return a new result. To use it, you simply specify the component and name of the property on whose binding you'd like to refresh.

Even though the function includes "db" in the name, the function can update all types of Vision component bindings, including Property and Expression bindings.

This function will only work within the Vision module. To manually execute bindings in Perspective, use the refreshBinding component method.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type.

Syntax

system.db.refresh(component, propertyName)

- Parameters

  JComponent component - The component whose property you want to refresh

  String propertyName - The name of the property that has a binding that needs to be refreshed

- Returns

  boolean - True (1) if the property was found and refreshed successfully.

- Scope

  Vision Client

Code Examples

Code Snippet - Refreshing a Table's Data Property

```
# This example could be placed in the actionPerformed event of a Button, to be used to
refresh the data of a Table.
# Remember to use the scripting name of the property that you're trying to refresh, and that
the property names are case-sensitive.

table = event.source.parent.getComponent("Table")
system.db.refresh(table, "data")
```

Keywords

system db refresh, db.refresh

system.db.removeDatasource

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

Description

Removes a database connection from Ignition.

Client Permission Restrictions

Permission Type: Datasource 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.db.removeDatasource(name)

- Parameters

  String name - The name of the database connection in Ignition.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Removing Database from Gateway

```
# This will result in the connection named MySQL being removed
system.db.removeDatasource("MySQL")
```

Keywords

system db removeDatasource, db.removeDatasource

system.db.rollbackTransaction

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

Description

Performs a rollback on the given connection. This will make all statements executed against this transaction since its beginning or since the last commit or rollback undone. Note that if you are done with the transaction, you must also close it after you do a rollback on it.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.db.rollbackTransaction(tx)

- Parameters

  String tx - The transaction ID.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Python - Rollback a Transaction on an Exception

```
# This example will use a for-loop to run multiple queries in a single Transaction, and
rollback if an error occurs.

# Create some variables for use later
txId = system.db.beginTransaction(timeout=5000)
status=2
query = "UPDATE MachineStatus SET status=? WHERE ID=?"
errors = False         # A flag to denote if we ran into a problem with a query during the
transaction

for machineId in range(8):
        try:
                system.db.runPrepUpdate(query,        args=[status, machineId],
tx=txId)
        except:
                errors = True
                break
# If we encountered an error...
if errors:
        # ...then rollback the transaction
        system.db.rollbackTransaction(txId)
else:
        # Otherwise, commit it
        system.db.commitTransaction(txId)
# In either case, close the transaction when we're done.
system.db.closeTransaction(txId)
```

Keywords

system db rollbackTransaction, db.rollbackTransaction

system.db.runNamedQuery

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

Description

Runs a named query and returns the results. **Note** that the number of parameters in the function is determined by **scope**. Both versions of the function are listed below.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Project Scope Syntax

system.db.runNamedQuery(path, parameters, [tx], [getKey])

- Parameters

  String path - The path to the named query to run. Note that this is the full path to the query, including any folders.

  PyDictionary parameters - A Python dictionary of parameters for the named query to use.

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

The following parameters were introduced in 8.0.3

  String tx - An optional transaction ID, obtained from beginNamedQueryTransaction. If blank, will not be part of a transaction.

  Boolean getKey - Optional. Only used for Update Query types. A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys.

- Returns

  Object - The results of the query. The exact object returned depends on the Query Type property of the Named Query: typically either a dataset when set to **Query**, an integer representing the number of rows affected when set to **Update Query**, or an object matching the datatype of the value returned by a **Scalar Query**.

- Scope

  Vision Client, Perspective Session

Gateway Scope Syntax

system.db.runNamedQuery(project, path, parameters, [tx], [getKey])

- Parameters

    String project - The project name the query exists in.

    String path - The path to the named query to run. Note that this is the full path to the query, including any folders.

    PyDictionary parameters - A Python dictionary of parameters for the named query to use.

> The following feature is new in Ignition version **8.0.3**
> Click here to check out the other new features

The following parameters were introduced in 8.0.3

    String tx - An optional transaction ID, obtained from beginNamedQueryTransaction. If blank, will not be part of a transaction.

    Boolean getKey - Optional. Only used for Update Query types. A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys.

- Returns

    Object - The results of the query. The exact object returned depends on the Query Type property of the Named Query: typically either a dataset when set to **Query**, an integer representing the number of rows affected when set to **Update Query**, or an object matching the datatype of the value returned by a **Scalar Query**.

- Scope

    Gateway

Code Examples

Simple Example - Without Parameters

```
# This example will run a Named Query without any parameters in the Project scope.
# The second argument in the function is NOT optional, so named queries that do not require a
parameter must still pass an empty dictionary as an argument.

# Request the Named Query with an empty dictionary as the second parameter.
system.db.runNamedQuery("folderName/myNamedQuery", {})
```

Gateway Scope Example

```
# This example will run a Named Query without any parameters in the Gateway scope.
# The last argument in the function is NOT optional, so named queries that do not require a
parameter must still pass an empty dictionary as an argument.

# Request the Named Query to execute.
system.db.runNamedQuery("ProjectName", "folderName/myNamedQuery", {})
```

Simple Example - With Parameters

```
# This example will run a Named Query while passing some parameters in the Project scope.
# The Named Query is assumed to have two parameters already defined on the Named Query:
# param1 : A string
# param2 : An integer

# Create a python dictionary of parameters to pass
parameters = {"param1":"my string", "param2":10}

# Run the Named Query
system.db.runNamedQuery("myUpdateQuery", parameters)
```

Keywords

system db runNamedQuery, db.runNamedQuery

system.db.runPrepQuery

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

Description

Runs a prepared statement against the database, returning the results in a PyDataSet.. Prepared statements differ from regular queries in that they can use a special placeholder, the question-mark character (?) in the query where any dynamic arguments would go, and then use an array of values to provide real information for those arguments. Make sure that the length of your argument array matches the number of question-mark placeholders in your query.

This call should be used for SELECT queries. This is a useful alternative to system.db.runQuery because it allows values in the WHERE clause, JOIN clause, and other clauses to be specified without having to turn those values into strings. This is safer because it protects against a problem known as a SQL injection attack, where a user can input data that affects the query's semantics.

The "?" placeholder refers to variables of the query statement that help the statement return the correct information. The "?" placeholder cannot reference column names, table names, or the underlying syntax of the query. This is because the SQL standard for handling the "?" placeholder excludes these items.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runPrepQuery(query, args, database, tx)

- Parameters

  String query - A query (typically a SELECT) to run as a prepared statement with placeholders (?) denoting where the arguments go.

  Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

  String database - The name of the database connection to execute against.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  PyDataSet - The results of the query as a PyDataSet.

- Scope

  Gateway

Syntax

system.db.runPrepQuery(query, args, database, tx)

- Parameters

  String query - A query (typically a SELECT) to run as a prepared statement with placeholders (?) denoting where the arguments go.

  Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

  String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  PyDataSet - The results of the query as a PyDataSet.

- Scope

  Vision Client, Perspective Session

Code Examples

Code Snippet - Running Prepared Query With Query Parameter

```
# This example would search for all records in a LogEntry table where the message contained a
user-entered search term.

search = event.source.parent.getComponent("SearchFor").text
# Wrap the term in % signs for LIKE-style matching
search = '%' + search + '%'

results= system.db.runPrepQuery("SELECT * FROM LogEntry WHERE EntryText LIKE ?", [search])
event.source.parent.getComponent("Table").data = results
```

Keywords

system db runPrepQuery, db.runPrepQuery

system.db.runPrepUpdate

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

Description

Runs a prepared statement against the database, returning the number of rows that were affected. Prepared statements differ from regular queries in that they can use a special placeholder, the question-mark character (?) in the query where any dynamic arguments would go, and then use an array of values to provide real information for those arguments. Make sure that the length of your argument array matches the number of question-mark placeholders in your query. This call should be used for UPDATE, INSERT, and DELETE queries.

This is extremely useful for two purposes:

- This method avoids the problematic technique of concatenating user input inside of a query, which can lead to syntax errors, or worse, a nasty security problem called a SQL injection attack . For example, if you have a user-supplied string that is used in a WHERE clause, you use single-quotes to enclose the string to make the query valid. What happens in the user has a single-quote in their text? Your query will fail. Prepared statements are immune to this problem.
- This is the only way to write an INSERT or UPDATE query that has binary or BLOB data. Using BLOBs can be very useful for storing images or reports in the database, where all clients have access to them.

The "?" placeholder refers to variables of the query statement that help the statement return the correct information. The "?" placeholder cannot reference column names, table names, or the underlying syntax of the query. This is because the SQL standard for handling the "?" placeholder excludes these items.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runPrepUpdate(query, args, database, [tx], [getKey], [skipAudit])

- Parameters

  String query - A query (typically an UPDATE, INSERT, or DELETE) to run as a prepared statement with placeholders (?) denoting where the arguments go.

  Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

  String database - The name of the database connection to execute against.

  String tx - Optional, A transaction identifier. If omitted, the update will be executed in its own transaction.

  Boolean getKey - Optional, A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys.

  Boolean skipAudit - Optional, A flag which, if set to true, will cause the prep update to skip the audit system. Useful for some queries that have fields which won't fit into the audit log.

- Returns

  Integer - The number of rows affected by the query, or the key value that was generated, depending on the value of the getKey flag.

- Scope

  Gateway

Syntax

system.db.runPrepUpdate(query, args, [database], [tx], [getKey], [skipAudit])

- Parameters

  String query - A query (typically an UPDATE, INSERT, or DELETE) to run as a prepared statement with placeholders (?) denoting where the arguments go.

  Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

  String database - Optional, The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - Optional, A transaction identifier. If omitted, the update will be executed in its own transaction.

  Boolean getKey - Optional, A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys.

  Boolean skipAudit - Optional, A flag which, if set to true, will cause the prep update to skip the audit system. Useful for some queries that have fields which won't fit into the audit log.

- Returns

  Integer - The number of rows affected by the query, or the key value that was generated, depending on the value of the getKey flag.

- Scope

  Vision Client, Perspective Session

Code Examples

Code Snippet - Inserting Data Into Database

```
# This example would gather some user entered text and insert it into the database.

userText = event.source.parent.getComponent("TextArea").text
userName = system.security.getUsername()
system.db.runPrepUpdate("INSERT INTO Comments (Name, UserComment) VALUES (?,?)", [userName,
userText])
```

Code Snippet - Inserting Data Into Database

```
# This example would gather some user entered text and insert it into the database.
# The difference between this example and the previous example is that this example is
explicitly declaring which database connection to run the query againt.
# Sometimes you need to run a query against a database connection that is not the default
connection.

userText = event.source.parent.getComponent("TextArea").text
userName = system.security.getUsername()
databaseConnection = "AlternateDatabase"
system.db.runPrepUpdate("INSERT INTO Comments (Name, UserComment) VALUES (?,?)", [userName,
userText], databaseConnection)
```

Code Snippet - Reading File as Bytes and Inserting Bytes Into Database

```
# This code would read a file and upload it to the database

filename = system.file.openFile() # Ask the user to open a file
if filename != None:
   filedata = system.file.readFileAsBytes(filename)
   system.db.runPrepUpdate("INSERT INTO Files (file_data) VALUES (?)", [filedata])
```

Code Snippet - Inserting Data and Retrieving the Number of Affected Rows Using getKey Parameter

```
# This example inserts a new user and gives it the 'admin' role. Demonstrates the ability to
retrieve a newly created key value.

# Get the username/password
name = event.source.parent.getComponent('Name').text
desc = event.source.parent.getComponent('Description').text
building = event.source.parent.getComponent('Building').selectedValue

# Insert the value
id = system.db.runPrepUpdate("INSERT INTO machines (machine_name, description) VALUES (?,
?)", [name, desc], getKey=1)

# Add a row to the user role mapping table
system.db.runPrepUpdate("INSERT INTO machine_building_mapping (machine_id, building) VALUES
(?, ?)", [id, building])
```

Code Snippet - Inserting Data From a Table Component

```
# This example will take a dataset from a table component, and insert new records into the
database, one row at a time


# Read the contents of the table
tableData = event.source.parent.getComponent('Table').data

# Convert it to a PyDataset. This is mostly for convenience, as they're easier to iterate
through
pyData = system.dataset.toPyDataSet(tableData)

# Build the query we'll use. You could easily modify the line to accommodate the table you're
trying to insert into.
query = "INSERT INTO my_table (col1, col2) VALUES (?, ?)"

# Iterate
for row in pyData:

        # Build an arguments list based on the current row. Using indexing here, so 'row[0]'
is the 1st column, 'row[1]' is the 2nd column, etc
        args = [row[0], row[1]]

        # Add a row to the database. You could optionally check the contents of the row
first, and add an if-statement to prevent the record based on some criteria
        system.db.runPrepUpdate(query, args)
```

Keywords

system db runPrepUpdate, db.runPrepUpdate

system.db.runQuery

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

Description

Runs a SQL query, usually a SELECT query, against a database, returning the results as a dataset. If no database is specified, or the database is the empty-string "", then the current project's default database connection will be used. The results are returned as a PyDataSet, which is a wrapper around the standard dataset that is convenient for scripting.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runQuery(query, database, tx)

- Parameters

  String query - A SQL query, usually a SELECT query, to run.

  String database - The name of the database connection to execute against.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  PyDataSet - The results of the query as a PyDataSet.

- Scope

  Gateway

Syntax

system.db.runQuery(query, database, tx)

- Parameters

  String query - A SQL query, usually a SELECT query, to run.

  String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  PyDataSet - The results of the query as a PyDataSet.

- Scope

  Vision Client, Perspective Session

Code Examples

Assuming the following dataset:

ID	Value
1	3.55
2	67.2
3	9.87

If you executed the following code:

Code Snippet

```
table = system.db.runQuery("SELECT * FROM TEST")
```

Table[2] would access the third row (rows are zero-indexed), and both table[2][0] and table[2]["ID"] would access the ID value of the third row.

As further example of how to use the results of runQuery, here are seven different ways to print out the table, and their results follow. Note that some of the later methods exercise some more advanced Jython concepts such as list comprehensions and string formatting, but their intent should be obvious. Generally speaking, the more concise Jython code becomes, the more readable it is.

Code Snippet - Executing Query and Printing Its Results

```
table = system.db.runQuery("SELECT * FROM Test")

print "Printing TEST Method 1..."
for row in table:
   for col in row:
      print col,
   print ""
print ""

print "Printing TEST Method 2..."
for row in table:
   print row[0], row[1]
print ""

print "Printing TEST Method 3..."
for row in table:
   print row["ID"], row["VALUE"]
print ""

print "Printing TEST Method 4..."
for rowIdx in range(len(table)):
   print "Row ",str(rowIdx)+": ", table[rowIdx][0], table[rowIdx][1]
print ""

print "Printing TEST Method 5..."
print [str(row[0])+", "+ str(row[1]) for row in table]
print ""

print "Printing TEST Method 6..."
print ["%s, %s" % (row["ID"],row["VALUE"]) for row in table]
print ""

print "Printing TEST Method 7..."
print [[col for col in row] for row in table]
print ""
```

The result would be:

```
Printing TEST Method 1...

0 3.55
```

```
1 67.2

2 9.87


Printing TEST Method 2...

0 3.55

1 67.2

2 9.87


Printing TEST Method 3...

0 3.55

1 67.2

2 9.87


Printing TEST Method 4...

Row 0: 0 3.55

Row 1: 1 67.2

Row 2: 2 9.87


Printing TEST Method 5...

['0, 3.55', '1, 67.2', '2, 9.87']


Printing TEST Method 6...

['0, 3.55', '1, 67.2', '2, 9.87']


Printing TEST Method 7...

[[0, 3.55], [1, 67.2], [2, 9.87]]
```

Keywords
system db runQuery, db.runQuery

system.db.runScalarPrepQuery

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

Description

Runs a prepared statement against a database connection just like the runPrepQuery function, but only returns the value from the first row and column. If no results are returned from the query, the special value None is returned.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runScalarPrepQuery(query, args, database, tx)

- Parameters

  String query - A SQL query (typically a SELECT) to run as a prepared statement with placeholders (?) denoting where the arguments go, that should be designed to return one row and one column.

  Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

  String database - The name of the database connection to execute against.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  Object - The value from the first row and first column of the results. Returns None if no rows were returned.

- Scope

  Gateway

Syntax

system.db.runScalarPrepQuery(query, args, database, tx)

- Parameters

  String query - A SQL query (typically a SELECT) to run as a prepared statement with placeholders (?) denoting where the arguments go, that should be designed to return one row and one column.

  Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

  String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  Object - The value from the first row and first column of the results. Returns None if no rows were returned.

- Scope

  Vision Client, Perspective Session

Code Examples

Code Snippet - Executing Query

```
# This example would search for the user id of someone based on a typed in username.
name = event.source.parent.getComponent("User Search").text

result = system.db.runScalarPrepQuery("SELECT user_id FROM users WHERE username = ?", [name])
event.source.parent.getComponent("Text Field").data = result
```

Keywords
system db runScalarPrepQuery, db.runScalarPrepQuery

system.db.runScalarQuery

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

Description

Runs a query against a database connection just like the runQuery function, but only returns the value from the first row and column. If no results are returned from the query, the special value None is returned.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runScalarQuery(query, database, tx)

- Parameters

  String query - A SQL query that should be designed to return one row and one column.

  String database - The name of the database connection to execute against.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  Object - The value from the first row and first column of the results. Returns None if no rows were returned.

- Scope

  Gateway

Syntax

system.db.runScalarQuery(query, database, tx)

- Parameters

  String query - A SQL query that should be designed to return one row and one column.

  String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - A transaction identifier. If omitted, the query will be executed in its own transaction.

- Returns

  Object - The value from the first row and first column of the results. Returns None if no rows were returned.

- Scope

  Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would count the number of active alarms, and acknowledge them all if there is at
least one.
numAlarms = system.db.runScalarQuery("SELECT COUNT(*) FROM alarmstatus " + "WHERE
unacknowledged = 1")
if numAlarms > 0:
   # There are alarms - acknowledge all of them
   system.db.runUpdateQuery("UPDATE alarmstatus SET unacknowledged = 0")
```

Code Snippet

```
# This code would read a single value from a table and show it to the user an a popup box.
level = system.db.runScalarQuery("SELECT Level FROM LakeInfo WHERE LakeId='Tahoe'")
system.gui.messageBox("The lake level is: %d feet" % level)
```

Keywords

system db runScalarQuery, db.runScalarQuery

system.db.runSFNamedQuery

The following feature is new in Ignition version **8.0.11**
Click here to check out the other new features

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

Description

Runs a named query that goes through the Store and Forward system. Note that the number of parameters in the function is determined by **scope**. Both versions of the function are listed on this page.

Only Update named queries are allowed in Store and Forward.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Gateway Scope Syntax

system.db.runSFNamedQuery(project, path, parameters, [getKey])

- Parameters

  String project - The project name the query exists in.

  String path - The path to the named query to run. Note that this is the full path to the query, including any folders.

  PyDictionary parameters - A Python dictionary of parameters for the named query to use.

- Returns

  Boolean - Returns true if successfully sent to the Store and Forward system.

- Scope

  Gateway, Perspective Session

Project Scope Syntax

system.db.runSFNamedQuery(path, parameters, [getKey])

- Parameters

  String path - The path to the named query to run. Note that this is the full path to the query, including any folders.

  PyDictionary parameters - A Python dictionary of parameters for the named query to use.

- Returns

  Boolean - Returns true if successfully sent to the Store and Forward system.

- Scope

  Vision Client

Code Examples

Simple Example - Without Parameters

```
# This example will run a Named Query without any parameters in the Project scope.
# The second argument in the function is NOT optional, so named queries that do not require a
parameter
# must still pass an empty dictionary as an argument.

# Request the Named Query with an empty dictionary as the second parameter.
system.db.runSFNamedQuery("folderName/myNamedQuery", {})
```

Gateway Scope Example

```
# This example will run a Named Query without any parameters in the Gateway scope.
# The last argument in the function is NOT optional, so named queries that do not require a parameter
# must still pass an empty dictionary as an argument.

# Request the Named Query to execute.
system.db.runSFNamedQuery("ProjectName", "folderName/myNamedQuery", {})
```

Simple Example - With Parameters

```
# This example will run a Named Query while passing some parameters in the Project scope.
# The Named Query is assumed to have two parameters already defined on the Named Query:
# param1 : A string
# param2 : An integer

# Create a python dictionary of parameters to pass
params = {"param1":"my string", "param2":10}

# Run the Named Query
system.db.runSFNamedQuery("myUpdateQuery", params)
```

Keywords
system db runSFNamedQuery, db.runSFNamedQuery

system.db.runSFPrepUpdate

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

Description

Runs a prepared statement query through the store and forward system and to multiple datasources at the same time. Prepared statements differ from regular queries in that they can use a special placeholder, the question-mark character (?) in the query where any dynamic arguments would go, and then use an array of values to provide real information for those arguments. Make sure that the length of your argument array matches the number of question-mark placeholders in your query. This call should be used for UPDATE, INSERT, and DELETE queries.

This is extremely useful for two purposes:

- This method avoids the problematic technique of concatenating user input inside of a query, which can lead to syntax errors, or worse, a nasty security problem called a SQL injection attack. For example, if you have a user-supplied string that is used in a WHERE clause, you use single-quotes to enclose the string to make the query valid. What happens in the user has a single-quote in their text? Your query will fail. Prepared statements are immune to this problem.
- This is the only way to write an INSERT or UPDATE query that has binary or BLOB data. Using BLOBs can be very handy for storing images or reports in the database, where all clients have access to them.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runSFPrepUpdate(query, args, datasources)

- Parameters

    String query - A query (typically an UPDATE, INSERT, or DELETE) to run as a prepared statement, with placeholders (?) denoting where the arguments go.

    Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

    String[] datasources - List of datasources to run the query through.

- Returns

    boolean - Returns true if successfully sent to store-and-forward system.

- Scope

    Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Run through single datasource
print system.db.runSFPrepUpdate("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES (?,?,?,?)",
['A Name', 1032, 234, 1], datasources=["MySQLDatasource"])
```

Code Snippet

```
# Example 2: Run through two datasources
print system.db.runSFPrepUpdate("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES (?,?,?,?)",
['A Name', 1032, 234, 1], datasources=["MySQLDatasource", "SQLServerDatasource"])
```

Keywords
system db runSFPrepUpdate, db.runSFPrepUpdate

system.db.runSFUpdateQuery

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

Description

Runs a query through the store and forward system and to multiple datasources at the same time.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runSFUpdateQuery(query, datasources)

- Parameters

  String query - A query (typically an UPDATE, INSERT, or DELETE) to run.

  String[] datasources - List of datasources to run the query through.

- Returns

  Boolean - Returns true if successful and false if not.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Run through single datasource
print system.db.runSFUpdateQuery("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES ('A Name',
1032, 234, 1)", ["MySQLDatasource"])
```

Code Snippet

```
# Example 2: Run through 2 datasources
print system.db.runSFUpdateQuery("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES ('A Name',
1032, 234, 1)", ["MySQLDatasource", "SQLServerDatasource"])
```

Keywords

system db runSFUpdateQuery, db.runSFUpdateQuery

system.db.runUpdateQuery

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

Description

Runs a query against a database connection, returning the number of rows affected. Typically this is an UPDATE, INSERT, or DELETE query. If no database is specified, or the database is the empty-string "", then the current project's default database connection will be used.

Note that you may want to use the runPrepUpdate query if your query is constructed with user input (to avoid the user's input from breaking your syntax) or if you need to insert binary or BLOB data.

Client Permission Restrictions

Permission Type: Legacy Database Access

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.db.runUpdateQuery(query, database, tx, getKey, skipAudit)

- Parameters

  String query - A SQL query, usually an INSERT, UPDATE, or DELETE query, to run.

  String database - The name of the database connection to execute against.

  String tx - A transaction identifier. If omitted, the update will be executed in its own transaction.

  Boolean getKey - A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys.

  Boolean skipAudit - A flag which, if set to true, will cause the update query to skip the audit system. Useful for some queries that have fields which won't fit into the audit log.

- Returns

  Integer - The number of rows affected by the query, or the key value that was generated, depending on the value of the getKey flag.

- Scope

  Gateway

Syntax

system.db.runUpdateQuery(query, database, tx, getKey, skipAudit)

- Parameters

  String query - A SQL query, usually an INSERT, UPDATE, or DELETE query, to run.

  String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used.

  String tx - A transaction identifier. If omitted, the update will be executed in its own transaction.

  Boolean getKey - A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys.

  Boolean skipAudit - A flag which, if set to true, will cause the update query to skip the audit system. Useful for some queries that have fields which won't fit into the audit log.

- Returns

  Integer - The number of rows affected by the query, or the key value that was generated, depending on the value of the getKey flag.

- Scope

  Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would acknowledge all unacknowledged alarms # and show the user how many alarms
were acknowledged.
rowsChanged = system.db.runUpdateQuery("UPDATE alarmstatus SET unacknowledged = 0")
system.gui.messageBox("Acknowledged %d alarms" % rowsChanged)
```

Code Snippet

```
# This example inserts a new user and gives it the 'admin' role.  Demonstrates the ability to
retrieve a newly created key value.
# get the username/password
name = event.source.parent.getComponent('Name').text
desc = event.source.parent.getComponent('Description').text
building = event.source.parent.getComponent('Building').selectedValue

# insert the value
id = system.db.runUpdateQuery("INSERT INTO machines (machine_name, description) " + "VALUES
('%s', '%s')" %(name, desc), getKey=1)

# add a row to the user role mapping table
system.db.runUpdateQuery("INSERT INTO machine_building_mapping " + "(machine_id, building)
VALUES (%d, %d)" %(id, building))
```

Keywords

system db runUpdateQuery, db.runUpdateQuery

system.db.setDatasourceConnectURL

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

Description

Changes the connect URL for a given database connection.

Client Permission Restrictions

Permission Type: Datasource 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.db.setDatasourceConnectURL(name, connectUrl)

- Parameters

  String name - The name of the database connection in Ignition.

  String connectUrl - The new connect URL.

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1:

system.db.setDatasourceConnectURL("MySQL", "jdbc:mysql://localhost:3306/test")
```

Keywords

system db setDatasourceConnectURL, db.setDatasourceConnectURL

system.db.setDatasourceEnabled

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

Description

Enables/disables a given database connection.

Client Permission Restrictions

Permission Type: Datasource 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.db.setDatasourceEnabled(name, enabled)

- Parameters

  String name - The name of the database connection in Ignition.

  Boolean enabled

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Enabling a Database Connection

```
# Enable the database named "MySQL"

system.db.setDatasourceEnabled("MySQL", 1)
```

Code Snippet - Disabling a Database Connection

```
# Disable the database named "MySQL"

system.db.setDatasourceEnabled("MySQL", 0)
```

Keywords

system db setDatasourceEnabled, db.setDatasourceEnabled

system.db.setDatasourceMaxConnections

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

Description

Sets the *Max Active* and *Max Idle* parameters of a given database connection.

Client Permission Restrictions

Permission Type: Datasource 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.db.setDatasourceMaxConnections(name, maxConnections)

- Parameters

  String name - The name of the database connection in Ignition.

  Integer maxConnections

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Setting the Max Connections of a Data Source

```
# Set the max connection count for the "MySQL" database to 20
system.db.setDatasourceMaxConnections("MySQL", 20)
```

Keywords

system db setDatasourceMaxConnections, db.setDatasourceMaxConnections

system.device

Device Functions

The following functions give you access to view and edit device connections in the Gateway.

In This Section ...

system.device.addDevice

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

Description

Adds a new device connection in Ignition. Accepts a dictionary of parameters to configure the connection. Acceptable parameters differ by device type: i.e., a Modbus/TCP connection requires a hostname and port, but a simulator doesn't require any parameters.

When using this function, the arguments *must* be passed in as keyword arguments.

Client Permission Restrictions

Permission Type: Device 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.device.addDevice(deviceType, deviceName, deviceProps)

- Parameters

  String deviceType - The device driver type. Possible values are listed in the Device Types table below.

  String deviceName - The name that will be given to the the new device connection.

  Dictionary deviceProps - A dictionary of device connection properties and values. Each deviceType has different properties, but most require at least a hostname. Keys in the dictionary are case-insensitive, spaces are omitted, and the names of the properties that appear when manually creating a device connection.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Device Types

The tables below represent Inductive Automation device types that can be created with this function. Some device types require manual configurations to become fully functional, such as loading configuration files or adding mapped entries. In these cases you won't be able to completely configure the device with this function alone. Those device types are marked with "(requires manual configuration)" in the table below.

In addition, this function can also add devices from third party modules; you will need to supply the driver type, which the module developer will be able to provide.

Driver Name	Device Type
Allen-Bradley Logix Driver	LogixDriver
Allen-Bradley MicroLogix	MicroLogix
Allen-Bradley PLC5	PLC5
Allen-Bradley SLC	SLC
DNP3 Driver	Dnp3Driver
Legacy Allen-Bradley CompactLogix	CompactLogix
Legacy Allen-Bradley ControlLogix	ControlLogix
Modbus RTU	ModbusRtuOverTcp
Modbus TCP	ModbusTcp
Omron FINS TCP (requires manual configuration)	com.inductiveautomation.FinsTcpDeviceType
Omron FINS UDP (requires manual configuration)	com.inductiveautomation.FinsUdpDeviceType
Omron NJ Driver (requires manual configuration)	com.inductiveautomation.omron.NjDriver

Editor notes are only visible to logged in users
This addition is on hold until issues are worked out in development. See IGN-1100 for progress.

Driver Name	Device Type
BACnet (requires manual configuration)	com.inductiveautomation.BacnetIpDeviceType

Driver Name	Device Type
Siemens S7-300	S7300
Siemens S7-400	S7400
Siemens S7-1200	S71200
Siemens S7-1500	S71500
Simulators Dairy Demo Simulator	DairyDemoSimulator
Simulators Generic Simulator	Simulator
Simulators SLC Simulator	SLCSimulator
TCP Driver	TCPDriver
UDP Driver	UDPDriver

Device Properties

The deviceProps parameter is where you supply configuration values to the new connection. Value properties depend on which deviceType was specified. A listing of deviceProps keys can be found on the system.device.addDevice - deviceProps Listing page.

The keys in the deviceProps parameter are case-insensitive. Device properties not specified in the deviceProps parameter will fallback to default values if not specified (where applicable: i.e., "hostname" typically does not have a default value).

Code Examples

Code Snippet - Creating a New Simulator Device

```
# This example creates a new Generic Simulator device connection.
# Note that we MUST pass a dictionary as the third parameter, even if it's empty.

# Call the function
system.device.addDevice(deviceType = "Simulator", deviceName = "New_Generic_Simulator",
deviceProps = {} )
```

Code Snippet - Creating a New Allen Bradley Logix Device

```
# Add a device using the Allen-Bradley Logix Driver for firmware v21+ devices
deviceProps = {}
deviceProps["Hostname"] = "192.168.1.2"
system.device.addDevice(deviceName="Test1", deviceType="LogixDriver", deviceProps=deviceProps)
```

Code Snippet - Creating a New Siemens Device

```
# This example creates a new S7-1500 device connection.

# Build a dictionary of parameters.
newProps = {
                        "HostName" : "10.0.0.1",
                        "Port" : 102 # <---If adding additional parameters, make sure to add
a comma.
                        }

# Call the function.
system.device.addDevice(deviceType = "S71500", \
                                                deviceName = "My_S7_1500_Device",\
                                                deviceProps = newProps )
```

Keywords
system device addDevice, device.addDevice

system.device.addDevice - deviceProps Listing

Description

Below is a table of properties callable by system.device.addDevice.

Note that the Description and Enabled properties may not be configured with this function, although a device connection could be disabled with a call to system.device.setDeviceEnabled() after creating the connection.

LogixDriver Keys

Device Property	Key
Hostname	hostname
Port	port
Timeout	timeout
Max Concurrent Requests	concurrency
Slot Number	slotnumber
Connection Path	path
Automatic Rebrowse	automaticrebrowseenabled
CIP Connection Size	cipconnectionsize

CompactLogix Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Connection Path	path
Concurrent Requests	concurrentRequests
Disable Automatic Browse	disableAutomaticBrowse
Show String Arrays	showStringArrays
Status Request Poll Rate	pollRate

Editor notes are only visible to logged in users
This addition is on hold until issues are worked out in development. See IGN-1100 for progress.

com.inductiveautomation.BacnetIpDeviceType

Device Property	Key
Local Device	
Remote Address	remote_address
Remote Port	remote_port
Remote Device Number	remote_device_number
Write Priority	write_priority
COV Enabled	cov_enabled
COV Heartbeat Interval	cov_heartbet_interval
COV Subscription Lifetime	cov_subscription_lifetime
Confirmed Notifications Enabled	confirmed_notifications_enabled

com.inductiveautomation.omron.NjDriver Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Concurrency	concurrency
Connection Size	connectionSize
Slot Number	slotNumber

com.inductiveautomation.FinsTcpDeviceType

Device Property	Key
Hostname	hostname
Port	port
Timeout	timeout
Local Address	localAddress
Source Network	sourceNetwork
Source Node	sourceNode
Source Unit	sourceUnit
Destination Network	destinationNetwork
Destination Nodes	destinationNode
Destination Unit	destinationUnit
Concurrent Requests	concurrentRequest
Max Request Size	maxRequestSize
Max Gap Size	maxGapSize
Write Priority Ratio	writePriorityRatio

com.inductiveautomation.FinsUdpDeviceType

Device Property	Key
Bind Address	bindAddress
Bind Port	bindPort
Remote Address	remoteAddress
Remote Port	remotePort
Timeout	timeout
Source Network	sourceNetwork
Source Node	sourceNode
Source Unit	sourceUnit
Destination Network	destinationNetwork
Destination Nodes	destinationNode
Destination Unit	destinationUnit
Concurrent Requests	concurrentRequest
Max Request Size	maxRequestSize
Max Gap Size	maxGapSize
Write Priority Ratio	writePriorityRatio

ControlLogix Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Connection Path	path
Concurrent Requests	concurrentRequests
Disable Automatic Browse	disableAutomaticBrowse
Show String Arrays	showStringArrays
Status Request Poll Rate	pollRate
Slot Number	slotNumber

Dnp3Driver Keys

Device Property	Key	Acceptable Values
Hostname	hostname	
Port	port	
Source Address	sourceAddress	
Destination Address	destinationAddress	
Integrity Poll Interval	integrityPollInterval	
Direct Operate Enabled	directOperateEnabled	
Unsolicited Messages Enabled	unsolicitedMessagesEnabled	
Message Fragment Size	maxMessageFragmentSize	
Message Timeout	timeout	
Retries	retries	
Link Layer Confirmation	linkLayerConfirmationEnabled	
Default Outstation Conformance Level	outstationConformanceDefault	• `"UNKNOWN"` • `"ONE"` • `"TWO"` • `"THREE"` • `"FOUR"`
Analog Input Points	analogInputDefaultValueType	• `"INTEGER"` • `"SHORT"` • `"FLOAT"` • `"DOUBLE"` • `"VARIATION_0"`

Analog Input Frozen Points	analogInputFrozenDefaultValueType	• "INTEGER" • "SHORT" • "FLOAT" • "DOUBLE" • "VARIATION_0"
Analog Output Points	analogOutputDefaultValueType	• "INTEGER" • "SHORT" • "FLOAT" • "DOUBLE" • "VARIATION_0"
Counter Points	counterDefaultValueType	• "INTEGER" • "SHORT" • "VARIATION_0"
Counter Frozen Points	counterFrozenDefaultValueType	• "INTEGER" • "SHORT" • "VARIATION_0"
Binary Input Points	binaryInputDefaultValueType	• "PACKED" • "WITH_FLAGS" • "VARIATION_0"
Double-Bit Binary Input Points	doubleBitBinaryInputDefaultValueType	• "PACKED" • "WITH_FLAGS" • "VARIATION_0"
Binary Output Points	binaryOutputDefaultValueType	• "PACKED" • "WITH_FLAGS" • "VARIATION_0"

MicroLogix Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Browse Cache Timeout	browseCacheTimeout
Connection Path	path
Disable Processor Browse	disableProcessorBrowse
Zero TNS Connection	useZeroTnsConnections

ModbusRtuOverTcp and ModbusTcp Keys

Device Property	Key
Hostname	hostname
Port	port
Communication Timeout	communicationTimeout
Max Holding Registers Per Request	maxHoldingRegistersPerRequest
Max Input Registers Per Request	maxInputRegistersPerRequest
Max Coils Per Request	maxCoilsPerRequest
Max Discrete Inputs Per Request	maxDiscreteInputsPerRequest
Reverse Word Order	reverseWordOrder
One-based Addressing	zeroBasedAddressing
Span Gaps	spanGaps
Allow Write Multiple Registers Request	writeMultipleRegistersRequestAllowed
Force Multiple Register Writes	forceMultipleRegisterWritesEnabled
Allow Write Multiple Coils Request	writeMultipleCoilsRequestAllowed
Allow Read Multiple Registers Request	readMultipleRegistersRequestAllowed
Allow Read Multiple Coils	readMultipleCoilsAllowed
Allow Read Multiple Discrete Inputs	readMultipleDiscreteInputsAllowed
Reconnect After Consecutive Timeouts	reconnectAfterConsecutiveTimeouts
Max Retry Count	maxRetryCount
Reverse String Byte Order	reverseStringByteOrder
Right Justify String	rightJustifyStrings
Read Raw Strings	readRawStrings

PLC5 Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Browse Cache Timeout	browseCacheTimeout
Connection Path	path
Disable Processor Browse	disableProcessorBrowse
Zero TNS Connection	useZeroTnsConnections

S7300, S7400, S71200, and S71500 Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Port	port
PDU Size	pduSize
Rack Number	rackNumber
CPU Slot Number	cpuSlotNumber
Reconnect After Consecutive Timeouts	reconnectAfterConsecutiveTimeouts

SLC Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Browse Cache Timeout	browseCacheTimeout
Connection Path	path
Disable Processor Browse	disableProcessorBrowse
Zero TNS Connection	useZeroTnsConnections

TCPDriver Keys

Device Property	Key	Acceptable Values
Port(s)	ports	
Address	address	
Inactivity Timeout	inactivityTimeout	
Message Delimiter Type	messageDelimiterType	• `"PacketBased"` • `"CharacterBased"` • `"FixedSize"`
Message Delimiter	messageDelimiter	
Field Count	fieldCount	
Field Delimiter	fieldDelimiter	
Writeback Enabled	writebackEnabled	
Writeback Message Delimiter	writebackDelimiter	

UDPDriver Keys

Device Property	Key	Acceptable Values
Port(s)	ports	
Address	address	
Message Delimiter Type	messageDelimiterType	• `"PacketBased"` • `"CharacterBased"` • `"FixedSize"`
Message Delimiter	messageDelimiter	
Field Count	fieldCount	
Field Delimiter	fieldDelimiter	
Message Buffer Size	messageBufferSize	
Multicast	multicast	

Keywords

system device addDevice, device.addDevice

system.device.listDevices

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

Description

Returns a dataset of information about each configured device. Each row represents a single device.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.device.listDevices()

- Parameters

  Nothing

- Returns

  Dataset - A dataset, where each row represents a device. Contains four columns: *Name*, *Enabled*, *State*, and *Driver*
  .

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Listing Devices Configured on Your Gateway

```
deviceDataset = system.device.listDevices()

# Assign the deviceDataset to a Power Table. This example assumes
# the Power Table is in the same container as the component that called this script.
event.source.parent.getComponent('Power Table').data = deviceDataset
```

Keywords

system device listDevices, device.listDevices

system.device.refreshBrowse

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

Description

Forces Ignition to browse the controller. Only works for Allen-Bradley controllers.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.device.refreshBrowse(deviceName)

- Parameters

  String deviceName - The name of the device in Ignition.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example:
system.device.refreshBrowse("CLX")
```

Keywords

system device refreshBrowse, device.refreshBrowse

system.device.removeDevice

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

Description

Removes a given device from Ignition.

Client Permission Restrictions

Permission Type: Device 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.device.removeDevice(deviceName)

- Parameters

  String deviceName - The name of the device in Ignition.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Removing a Device from the Gateway

```
# Example:
system.device.removeDevice("CLX")
```

Keywords

system device removeDevice, device.removeDevice

system.device.setDeviceEnabled

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

Description

Enables/disables a device in Ignition.

Client Permission Restrictions

Permission Type: Device 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.device.setDeviceEnabled(deviceName, enabled)

- Parameters

  String deviceName - The name of the device in Ignition.

  Boolean enabled

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Enable a device

system.device.setDeviceEnabled("CLX", 1)
```

Code Snippet

```
# Example 2: Disable a device

system.device.setDeviceEnabled("CLX", 0)
```

Keywords

system device setDeviceEnabled, device.setDeviceEnabled

system.device.setDeviceHostname

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

Description
Changes the hostname of a device. Used for all ethernet based drivers.

Client Permission Restrictions
Permission Type: Device 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.device.setDeviceHostname(deviceName, hostname)

- Parameters

  String deviceName - The name of the device in Ignition.

  String hostname - The new IP address or hostname.

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Changing Device Hostname

```
# Example 1:

system.device.setDeviceHostname("CLX", "10.10.1.20")
```

Keywords

system device setDeviceHostname, device.setDeviceHostname

system.dnp3

DNP3 Functions

The following functions give you access to interact with the DNP3 devices.

Constants

system.dnp3.NUL = 0
system.dnp3.PULSE_ON = 1
system.dnp3.PULSE_OFF = 2
system.dnp3.LATCH_ON = 3
system.dnp3.LATCH_OFF = 4
system.dnp3.CLOSE = 1
system.dnp3.TRIP = 2

Status Codes

Many of the dnp3 functions return a status code. Those codes and their meaning are listed below.

Code Number	Identifier Name	Description
0	SUCCESS	Request accepted, initiated, or queued.
1	TIMEOUT	Request no accepted because the operate message was received after the arm timer timed out. The arm timer was started when the select operation for the same point was received.
2	NO_SELECT	Request no accepted because no previous matching select request exists. (An operate message was sent to activate an output that was not previously armed with a matching select message).
3	FORMAT_ERROR	Request not accepted because there were formatting errors in the control request (either select, operate, or direct operate).
4	NOT_SUPPORTED	Request not accepted because a control operation is not supported for this point.
5	ALREADY_ACTIVE	Request not accepted, because the control queue is full or the point is already active.
6	HARDWARE_ERROR	Request not accepted because of control hardware problems.
7	LOCAL	Request not accepted because Local/Remote switch is in Local position.
8	TOO_MANY_OBJS	Request not accepted because too many objects appeared in the same request.
9	NOT_AUTHORIZED	Request not accepted because of insufficient authorization.
10	AUTOMATION_INHIBIT	Request not accepted because it was prevented or inhibited by a local automation process.
11	PROCESSING_LIMITED	Request not accepted because the device cannot process any more activities than are presently in progress.
12	OUT_OF_RANGE	Request not accepted because the value is outside the acceptable range permitted for this point.
13 to 125	RESERVED	Reserved for future use.
126	NON_PARTICIPATING	Sent in request messages indicating that the outstation will not issue or perform the control operation.

127	UNDEFINED	Request not accepted because of some other undefined reason.

In This Section ...

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 int, short, float, or double).

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

- Returns

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

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example shows setting the analog output at index 0 to the
# Double value 3.14

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

Code Snippet

```
# This example shows setting the analog output at index 2 to the
# Integer value 300

system.dnp3.directOperateAnalog("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.directOperateAnalog("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.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.

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

- 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 shows setting 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. An empty list will freeze all analogs.

- 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.

- 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.freezCounters("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. An empty list will freeze all counters.

- 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.

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

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

- Returns

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

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example shows setting 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.

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

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

- 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 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

  none

- 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 will loop 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 will grab 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 = Errored

- Scope

  Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Querying Agent Status Information

```
# This script will loop 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.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 will grab status information from agents called Agent1, Agent2, Agent3, and will then place
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.**

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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

Permission Type: EAM Task Execution

This scripting function has 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

All

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 - Running an Agent Task

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

Code Snippet - Running an Agent Task and Seeing Its Response

```
# 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.getInfos()
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 (1) if the file/folder exists, false (0) 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.saveFile("")
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 - An extension, like ".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 temorary 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, like "pdf", to try to open. [optional]

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

- Returns

  String - The path to the selected file, or None if canceled.

- Scope

  Vision Client

Code Examples

Code Snippet - Opening a File

```
# This code would prompt 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 would prompt 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, like "pdf", to try to open. [optional]

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

- Returns

  List of Strings - The paths to the selected files, or None if canceled.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This code would prompt 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 would prompt 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 would prompt 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

  byte[] - The contents of the file as an array of bytes.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would prompt the user to choose a file. If the user chooses a file, it would then
read that file and upload it 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 would prompt the user to choose a text file. If the user chooses a file, it would
then set a text area on the screen 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, like "jpeg", for the file.

  String typeDesc - A description of the extension, like "JPEG Image".

- 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 would prompt 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(1), the file will be appended to if it already exists. If false(0), the file will be overwritten if it exists. The default is false(0). [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

  No return value

- 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.

  byte[] data - The binary content to write to the file.

  boolean append - If true(1), the file will be appended to if it already exists. If false(0), the file will be overwritten if it exists. The default is false(0). [optional]

- Returns

  No return value

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
#This code would download a BLOB from a database and save 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 would write 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.

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

- Returns

  none

- 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

  PySequence paths - A collection of paths to remove. The group paths are "Folder/Path/To/GroupName", separated by forward slashes.

- Returns

  none

- 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

  None

- Scope

  Vision Client

Code Snippet

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

Code Snippet

```
# The following example will close 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

  int red - The red component of the color, an integer 0-255.

  int green - The green component of the color, an integer 0-255.

  int blue - The blue component of the color, an integer 0-255.

  int 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 (1) if the user selected "Yes", false (0) 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 some 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

  int x - The X-coordinate, relative to the component that fired the event.

  int y - The Y-coordinate, relative to the component that fired the event.

  EventObject event - An event object for a component event.

- Returns

  PyTuple - A tuple of (x,y) in screen coordinates.

- Scope

  Vision Client

Code Examples

Code Snippet - Getting Location of Mouse Click

```
# This example will get the coordinates where the mouse is (from the corner of the monitor)
and display 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. Passing in a function of None will cause a separator line to appear in the popup menu, and the corresponding string will not be displayed. 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 mouseReleasedevents 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

  PySequence itemNames - A list of names to create popup menu items with.

  PySequence itemFunctions - A list of functions to match up with the names.

- Returns

  JPopupMenu - The javax.swing.JPopupMenu that was created.

- Scope

  Vision Client

Code Examples

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 a Linux or a Mac. 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 is a very basic to demonstrate 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

```
# 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 will make a message box appear on the Primary Desktop,
# regardless of where the script originates from.
# 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.gui.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 of the 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

  None

- 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.**

The following feature is new in Ignition version **8.0.6**
Click here to check out the other new features

Description

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

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.gui.getScreenIndex()

- Parameters

  None

- Returns

  Integer -The screen that the function was called from.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# Prints an integer representing the screen that the function was called from.
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. Exampe: If the original client opened 2 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

  None

- Returns

  PySequence - A 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, returning a tuple of their paths.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.gui.getOpenedWindowNames()

- Parameters

  None

- Returns

  PyTuple - 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, returning a tuple of references to them.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.gui.getOpenedWindows()

- Parameters

  None

- Returns

  PyTuple - 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, returning 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

  PyObject - 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

  int - 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 is
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

  PySequence - 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

  PyObject - Returns reference to the sibling component.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This example will get its sibling Text Field's text, and use 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

  PyObject - A reference to the window object, if it was open.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This example will get the window named 'Overview' and then close 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 will set 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

  None

- Returns

  PyTuple - A tuple of strings, representing the path of each window defined in the current project.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This example would open windows that begin with "Motor" and pass 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 press "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

```
# This could go in the mouseClicked event of a label to allow 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 touchscreen mode is currently enabled.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.gui.isTouchscreenModeEnabled()

- Parameters

  None

- Returns

  boolean - True(1) if the client currently has touchscreen mode activated.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This example should be 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 touchscreen 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.

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

  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.

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

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

  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.

  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.

  PySequence windows - A list of window paths to open in the new Desktop frame.

- 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

  None

- Returns

  None

- 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 would prompt 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 2nd 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 touchscreen 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 touchscreen 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 touchscreen 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.

  int 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 touchscreen 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 Jython 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 "typed".

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.

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

  boolean passwordMode - True (1) to activate password mode, where the text entered isn't echoed back clear-text. [optional]

- Returns

  String - The text that was "typed" 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 touchscreen 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 which are centered or have more than 2 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.

  int newX - An optional x-coordinate to move to, relative to the upper-left corner of the component's parent container.

  int newY - An optional y-coordinate to move to, relative to the upper-left corner of the component's parent container.

  int newWidth - An optional width for the component.

  int newHeight - An optional height for the component.

  int duration - An optional duration over which the transformation will take place. If omitted or 0, the transform will take place immediately.

  PyObject callback - An optional function to be called when the transformation is complete.

  int framesPerSecond - An optional frame rate argument which dictates how often the transformation updates over the given duration. The default is 60 frames per second.

  int acceleration - An optional modifier to the acceleration of the transformation over the given duration. See system.gui constants for valid arguments.

  int 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.

- Returns

  PyObject animation - An object that contains pause(), resume(), and cancel() methods, allowing for a script to interrupt the animation.

- Scope

  Vision Client

Code Examples

```
# This example changes the size the a component to 100x100
# This script should be run from the component that will be changed (ie: 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 pop-up 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 show 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')

# Cannot start the motor in mode #1
if runMode == 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

  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

  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 4 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

  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.

Syntax

system.math.mean(values)

- Parameters

  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

  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**.

  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

  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

  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

  Float[] values - A Sequence of numerical values. Accepts both Integers and Floats. The sequence may not contain None type values.

- Returns

  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

  Float[] values - A Sequence of numerical values. Accepts both Integers and Floats. The sequence may not contain None type values.

- Returns

  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

  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 through 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]

# Prints 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

  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]

# Prints 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

  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]

# Prints 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

  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]

# Prints 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

  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]

# Prints 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

  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]

# Prints 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

  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**.

  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]

# Prints 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

  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]

# Prints 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

  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]

# Prints 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

  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]

# Prints 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 an 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

```
# This code would be placed in the actionPerformed event of a button,
# and would close 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 would get the window named 'Overview' and then close it.
# If the window isn't open, show a warning
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 would close the window named 'Overview' in one step.
# If the window isn't open, the call to closeWindow will have 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. The screen index cast to 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. [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 will close 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 Strategy, 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

  none

- 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

  none

- Returns

  PyObject - 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

  none

- Returns

  PyObject - 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 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

  none

- Returns

  PyObject - 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" has a dynamic variable in its root container named "TankNumber", then callingsystem.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 Vision Project Properties restrictions.

Syntax

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

- Parameters

  String path - The path to the window to open.

  PyDictionary 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

  PyObject - 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.

The following feature is new in Ignition version **8.0.5**
Click here to check out the other new features

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.

  PyDictionary 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

  PyObject - A reference to the opened window. Refer to the list of window objects.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This example would open 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.

  PyDictionary 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

  PyObject - 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, or this will act like an openWindow.

  String swapToPath - The name of the window to swap to.

  PyDictionary 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

  PyObject - 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.

  PyDictionary 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

  PyObject - 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
# window containing the button and into a window named MyWindow
system.nav.swapWindow(event, "MyWindow")
```

Code Snippet

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

Code Snippet

```
# This code would swap 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.getExternalIpAddress

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 NAT router, then this address will be the WAN address of the router instead of the LAN address of the client, which is what you'd get with sy stem.net.getIpAddress.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.net.getExternalIpAddress()

- Parameters

  None

- 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.getExternalIpAddress()
# 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 getExternalIpAddress, net.getExternalIpAddress

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_WORKSTATION or bobs-laptop.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.net.getHostName()

- Parameters

  none

- 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

  none

- 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- [Optional] 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.

- Returns

  String[] - A List of Strings representing Gateway Network server ids.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# The following will create a list of running servers on the Gateway Network, and show 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

The following feature is new in Ignition version **8.0.6**
Click here to check out the other new features

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()

- Parameters

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

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

  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. [Optional. Defaults to None.]

  String password - A string indicating the password to use for authentication. [Optional. Defaults to None.]

  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. [Optional. Defaults to None.]

  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. [Optional. Defaults to "Normal".]

  - `"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. [Optional. Defaults to None.]

- 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 Response object, or a 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
`.get()`	Sends an HTTP GET call, blocking for a response.	Response
`.getAsync()`	Sends an HTTP GET call without blocking.	Promise
`.post()`	Sends an HTTP POST call, blocking for a response.	Response
`.postAsync()`	Sends an HTTP POST call without blocking.	Promise
`.put()`	Sends an HTTP PUT call, blocking for a response.	Response
`.putAsync()`	Sends an HTTP PUT call without blocking.	Promise
`.delete()`	Sends an HTTP DELETE call, blocking for a response.	Response
`.deleteAsync()`	Sends an HTTP DELETE call without blocking.	Promise
`.patch()`	Sends an HTTP PATCH call, blocking for a response.	Response
`.patchAsync()`	Sends an HTTP PATCH call without blocking.	Promise
`.head()`	Sends an HTTP HEAD call, blocking for a response.	Response
`.headAsync()`	Sends an HTTP HEAD call without blocking.	Promise
`.options()`	Sends an HTTP OPTIONS call, blocking for a response.	Response
`.optionsAsync()`	Sends an HTTP OPTIONS call without blocking.	Promise
`.trace()`	Sends an HTTP TRACE call, blocking for a response.	Response
`.traceAsync()`	Sends an HTTP TRACE call, without blocking for a response.	Promise
`.request()`	Sends an HTTP request, using a verb specified by the `method` parameter. Use this method in cases where a non-standard verb is required, and you need the call to block.	Response
`.requestAsync()`	Sends an HTTP request, with a verb specified by the `method` 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. [Optional. Defaults to None.]

- 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. [Optional. Defaults to None.]

- 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. [Optional. Defaults to None.]

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

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. [Optional. Defaults to None.]

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

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

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 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 (ie, 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()` `.cookieStore`	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.	Cookie Store
`.getCookieManager()` `.cookieManager`	Sends an HTTP GET call, blocking for a response.	Cookie Policy
`.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.	Byte Array

`.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()` `.clientError`	Returns True if the response was a client error, as in an HTTP 4XX response.	Boolean
`.isServerError()` `.serverError`	Returns True if the response was a server error, as in an HTTP 5XX response.	Boolean
`.getUrl()` `.url`	Returns the URL this Response connected to.	String
`.getHeaders()` `.headers`	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
`.getJavaResponse()` `.javaResponse`	Returns the underlying Java HttpResponse behind this Response.	HttpResponse

`.getCookieManager()` `.cookieManager`	Returns the CookieManager. See the CookieManager section for more details.	CookieManager
`.getRequest()` `.request`	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
`.get([timeout])`	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
`.then(callback)`	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
`.handleException(callback)`	In the event of an exception in a potential chain of promises, `handleException` 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.

Keep in mind that JRE proxy settings will influence how these functions conduct their network activities.

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 - [Optional] The MIME type used in the HTTP 'Content-type' header.

  Int connectTimeout - [Optional] The timeout for connecting to the URL in milliseconds. Default is 10,000

  Int readTimeout - [Optional] The read timeout for the operation in milliseconds. Default is 60,000.

  String username - [Optional] If specified, the call will attempt to authenticate with basic HTTP authentication.

  String password - [Optional] The password used for basic HTTP authentication, if the username parameter is also present.

  PyDictionary headerValues - [Optional] A dictionary of name/value pairs that will be set in the HTTP header.

  Boolean bypassCertValidation - [Optional] If the target address in 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.

- 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'll get the same thing you'd get by going to that website in a browser and using the browser's "View Source" function.

Keep in mind that JRE proxy settings will influence how these functions conduct their network activities.

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.

  Integer readTimeout - The read timeout for the get operation. In milliseconds. Default is 60,000.

  String username - If specified, the call will attempt to authenticate with basic HTTP authentication.

  String password - The password used for basic HTTP authentication, if the username parameter is also present.

  PyDictionary headerValues - Optional - A dictionary of name/value pairs that will be set in the HTTP header.

  Boolean bypassCertValidation - Optional - 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.

  Boolean useCaches - Optional - 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.

  Boolean throwOnError - Optional - 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.

- Returns

  String - The content found at the given URL.

- Scope

  Gateway, Vision Client, Perspective Session

If you are using JSON, check out the system.util.jsonEncode and system.util.jsonDecode functions.

Code Examples

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 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.

Keep in mind that JRE proxy settings will influence how these functions conduct their network activities.

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.

  PyDictionary 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 - Optional - The MIME type to use in the HTTP "Content-type" header.

  String postData - The raw data to post via HTTP.

  Integer connectTimeout - The timeout for connecting to the url. In milliseconds. Default is 10,000.

  Integer readTimeout - The read timeout for the get operation. In milliseconds. Default is 60,000.

  String username - If specified, the call will attempt to authenticate with basic HTTP authentication.

  String password - The password used for basic http authentication, if the username parameter is also present.

  PyDictionary headerValues - Optional - A dictionary of name/value pairs that will be set in the http header.

  Boolean bypassCertValidation - Optional - 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.

  Boolean throwOnError - Optional - 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.

- Returns

  String - The content returned for the POST operation.

- Scope

  All

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.

Keep in mind that JRE proxy settings will influence how these functions conduct their network activities.

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 - [Optional] The MIME type used in the HTTP 'Content-type' header.

  String putData - The raw data to put via HTTP.

  Int connectTimeout - [Optional] The timeout for connecting to the URL in milliseconds. Default is 10,000

  Int readTimeout - [Optional] The read timeout for the operation in milliseconds. Default is 60,000.

  String username - [Optional] If specified, the call will attempt to authenticate with basic HTTP authentication.

  String password - [Optional] The password used for basic HTTP authentication, if the username parameter is also present.

  PyDictionary headerValues - [Optional] A dictionary of name/value pairs that will be set in the HTTP header.

  Boolean bypassCertValidation - [Optional] If the target address in 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.

- 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.

  PyDictionary 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 make use of 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).

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 (1), 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)

- 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

  String body - The body text of the email.

  Boolean html - A flag indicating whether or not to send the email as an HTML email. Will auto-detect if omitted.

  String[] to - A list of email addresses to send to.

  String[] attachmentNames - A list of attachment names. Attachment names must have the correct extension for the file type or an error will occur.

  byte[][] attachmentData - A list of attachment data, in binary format.

  Integer timeout - A timeout for the email, specified in milliseconds. Defaults to 300,000 milliseconds (5 minutes).

  String username - If specified, will be used to authenticate with the SMTP host.

  String password - If specified, will be used to authenticate with the SMTP host.

  String priority - Priority for the message, from "1" to "5", with "1" being highest priority. Defaults to "3" (normal) priority.

  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.

  String[] cc - A list of email addresses to carbon copy. Only available if a smtpProfile is used.

  String[] bcc - A list of email addresses to blind carbon copy. Only available if a smtpProfile is used.

  Integer retries - The number of additional times to retry sending on failure. Defaults to 0. Only available if a smtpProfile is used.

  String[] replyTo - An optional list of addresses to have the recipients reply to. If omitted, this defaults to the from address.

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would send 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 would send 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 ask 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 would send 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 browse 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.

Caution: This function performs a fully recursive browse that can't be terminated, which can be especially problematic in larger systems. It is highly advised to use system.opc.browseServer instead since recursion with that function is driven by subsequent calls.

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, returns a list of OPCBrowseElement objects for the given server. Otherwise returns a list of OPCBrowseElements.

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

The PyOPCTag 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.

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 PyOPCTag 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.

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

  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 will print 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
for address in OpcObjects:
        print address.getOpcItemPath()
```

Keywords
system opc browseSimple, opc.browseSimple

system.opc.getServers

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.getServers([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. added in 8.0.14

- 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.getServers()
if not servers:
        print "No servers found"
else:
         for server in servers:
                print server
```

Keywords

system opc getServers, opc.getServers

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 will check the state of all configured servers, and show 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 will iterate 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.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 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/N11:0The 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 readValue 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.

  String[] itemPaths - A list of strings, each representing an item path, or address to read from.

- Returns

  QualifiedValue[] - A sequence of readValues objects, one for each address specified, in order. Each object will contains 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 associated 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

  No return value.

- 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.

  String[] itemPaths - A list of item paths, or addresses, to write to.

  Object[] values - A list of values to write to each address specified.

- Returns

  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

Code Snippet

```
# This example will write values to specified OPC items
# Declare a list of values
objectValue = [3]

# Declare the item path to write values back to
itemPath = "ns=1;s=[GoldSim]B3/B3:100"

# Write the values to the specified item. Replace "Ignition OPC UA Server" with your chosen
OPC UA server.
system.opc.writeValues("Ignition OPC UA Server", itemPath, objectValue)
```

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

  BrowseResults[] - The Browse Results that would result for the operation at that root. Refer to the list of Results objects.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no 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

  Aggregates[] - 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 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

  Attributes[] - 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.	int
getName()	Returns the name of the attribute.	String
getDesc()	Returns the description of the attribute.	String
getType()	Returns the datatype of the attribute.	Datatype
setType(type)	Sets the type on the attribute. Has one parameter, which is a DataType object.	Null
toString()	Returns a formatted string that lists the Id, Name, Description, and Datatype of the attribute.	String

Code Examples

There are no 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

  None

- Returns

  Names[] - A list of the string names of servers.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no 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 if the given item ID does not 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.insert(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 insert.

  Object date - The date to insert.

  int quality - The quality to insert.

- Returns

  QualityCode - The result of the insert.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

```
# This will insert 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

Will insert values on the OPC-HDA server, or replace 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 to perform the operation on.

  Object value - The value to insert or replace.

  Object date - The date to insert or replace.

  int quality - The quality to insert or replace.

- Returns

  QualityCode - The result of the insert or replace operation.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no 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 - Will be true if the server is available and can be queried, false if not.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no 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

  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 is itself a list of QualifiedValues.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no 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 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.

  int resampleIntervalMS - The interval, in milliseconds, that each value should cover.

  List aggregates - A list which should be one-toone 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

  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 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.

  int 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

  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 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.

  int quality - The quality to replace.

- Returns

  int - The items quality resulting from the operation.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no 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 NodeId of the Object Node the Method is a member of.

  String methodId - The NodeId 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

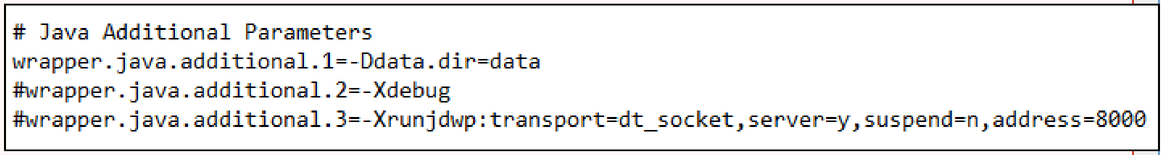
This function is used in **Python Scripting.**

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.

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=8000
```

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 the meet the remoteLevel logging level requirement. [Optional]

  String level - The desired Session logging level. Possible values are: all, trace, debug, info, warn, error, fatal, 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

  None

- 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

  None

- Scope

  Perspective Session

Code Examples

Code Snippet

```
# Will close a docked view with the given dock id.
system.perspective.closeDock('myDockID')
```

Keywords

system perspective closeDock, perspective.closeDock

system.perspective.closePage

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

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, then the pageId parameter must be included in the call. [optional]]

  String pageId - Identifier of the page to be closed. If omitted, the current pageId is used. [optional]

- Returns

  None

- 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

  None

- 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

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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

Description

Closes the Perspective Session with the given session ID 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.

In the Perspective mobile 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

  None

- 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. 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

  None

- 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.getSessionInfo

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

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:

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, "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.

This function accepts keyword arguments.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

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. [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 will get all sessions using the "MyProject" project and display 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

The following feature is new in Ignition version **8.0.2**
Click here to check out the other new features

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)

- 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.

  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.

- Returns

  True if the user in the current 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 login with the project's configured Identity Provider (IdP). For this function to work, an Identity Provider 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.

Note that calling this function after a user is already logged in will **not** log out the previous user.

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 altogether.

Editor notes are only visible to logged in users
Removed the "recommended" approach, since it doesn't seem to address the actual use case. Waiting to hear back from dev before we add something back.

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 Identity Provider to re-authenticate the user, even if the user is already signed into the Identity Provider. If set to true, then the Identity Provider will ask the user to re-enter their credentials. If set to false, then the Gateway will request that the Identity Provider 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 Identity Provider; 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

  None

- 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 must be set in the Perspective project properties.

⚠ 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 altogether.

Editor notes are only visible to logged in users
Hiding this. See the note on the system.perspective.login page for context.

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

  None

- Scope

  Perspective Session

Code Examples

There are no examples associated with this scripting 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

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.

  String view - If specified, will navigate to a specific view. Navigating to a view via 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.

  PyDictionary 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

  None

- 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.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, 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]

  PyDictionary params - Parameters that can be passed into the docked view. Must match the docked views View Parameters. Added in 8.0.1. [optional]

- Returns

  None

- 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])

- 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.

  PyDictionary params - Dictionary of key-value pairs to us 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]

  PyDictionary 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. [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]

- Returns

  None

- 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

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

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

Description

Sends print statements to the scripting 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.

  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]

  The following feature is new in Ignition version **8.0.10**
  Click here to check out the other new features

  String destination - Where the message should be printed. If specified, must be "client", "gateway", or "all". Default is "client". [optional]

- Returns

  None

- 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.

⚠ 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

  None

- Scope

  Perspective Session

Code Examples

There are no examples associated with this scripting 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 is passed "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 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

  None

- 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

  None

- 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]

  PyDictionary params - Parameters that can be passed into the docked view. Must match the docked views View Parameters. [optional]

- Returns

  None

- 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])

- 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.

  PyDictionary params - Dictionary of key-value pairs to us as input parameters to the View. [Optional]

  String title - Text to display in the title bar. [Optional]

  PyDictionary 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]

- Returns

  None

- Scope

  Perspective Session

Code Examples

Code Snippet - View

```
# Toggles a popup view. We are passing in two parameters, called "myParam" and "myParam2".
system.perspective.togglePopup("myPopupId",'folder/myView', params = {"myParam":1,"myParam2":"
Test"})
```

Keywords

system perspective togglePopup, perspective.togglePopup

system.perspective.vibrateDevice

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

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

Description

When called from the Perspective mobile app, will cause the device to vibrate for the specified number of milliseconds.

⚠ 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.

⚠ 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

  None

- 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.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
Advanced Function. Takes a snapshot of a component and creates a Java BufferedImage out of it. You can use javax.imageio.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
There are no examples associated with this scripting function.

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	.isShowPrintDialog()	.setShowPrintDialog(boolean)
Fit To Page	.isFitToPage()	.setFitToPage(boolean)
Zoom Factor	.getZoomFactor()	.setZoomFactor(double)
Orientation	.getOrientaion()	.setOrientation(int)
Page Width	.getPageWidth()	.setPageWidth(float)
Page Height	.getPageHeight()	.setPageHeight(float)
Left Margin	.getLeftMargin	.setLeftMargin(float)
Right Margin	.getRightMargin()	.setRightMargin(float)
Top Margin	.getTopMargin()	.setTopMargin(float)
Bottom Margin	.getBottomMargin()	.setBottomMargin(float)
Printer Name	.getPrinterName()	.setPrinterName(string)
All Margins*	-	.setMargins(float)

*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 objects.

- 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

The following feature is new in Ignition version **8.0.11**
Click here to check out the other new features

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

  none

- Returns

  String - The name of the currently running project.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would display 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

The following feature is new in Ignition version **8.0.11**
Click here to check out the other new features

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

  none

- 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 would print 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.

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 as of 8.0.5.

  PyDictionary parameters - An optional dictionary of parameter overrides, in the form name:value pairs.

  String action - The name of the distribution action to use. The action parameter supports the following keys as strings:

  - email
  - print
  - save
  - ftp

  PyDictionary actionSettings - An optional dictionary of settings particular to the action. Missing values will use the default value for that action.

- Returns

  None

- Throws

  IllegalArgumentException - Thrown 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.

- 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.

The email action now has the ability to add emails to the reply to field of the email. The *replyTo*, *replyToRoles*, and *re plyToUserSource* keys have been added to the possible dictionay 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, *p ageOrientation*, 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 filetype and format

The following is a list of values for the actionSettings filetype and format keys.

- pdf
- html
- csv
- rtf
- jpeg
- png
- xml

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

The following values for filetype were added in 8.0.4:

- xls
- xlsx

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 byte[] of the output.

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 client scope. Optional in session scope as of 8.0.5.

    PyDictionary parameters - A optional dictionary of parameter overrides, in the form name:value.

    String fileType - The file type the resulting byte array should represent. Defaults to "pdf". Not case-sensitive

- Returns

    byte[] - A byte array of the resulting report.

- Throws

    IllegalArgumentException - Thrown when any of the following occurs: If the file type is not recognized, path does not exist, project does not exist.

- Scope

    Gateway, Vision Client, Perspective Session

Values for filetype and format

The following is a list of values for the actionSettings filetype and format keys.

- pdf
- html
- csv
- rtf
- jpeg
- png
- xml

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

The following values for filetype were added in 8.0.4:

- xls
- xlsx

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. This dataset is particularly suited for display in a Tree View component

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 as of 8.0.5.

- Returns

  Dataset - A dataset of report paths and names for the project. Returns an empty dataset if the project has no reports.

- Throws

  IllegalArgumentException - Thrown when any of the following occurs: If the project name is omitted in the Gateway scope, project does not exist.

- 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.

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. Optional in session scope as of 8.0.5.

- Returns

  List - A list of report paths for the project. Returns an empty list if the project has no reports.

- Throws

  IllegalArgumentException - Thrown when any of the following occurs: If the project name is omitted in the Gateway scope, project does not exist.

- 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

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

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

  none

- 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

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

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

  none

- 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.getRosters

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

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

  none

- Returns

  Dictionary - 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 will print out all existing rosters to the console of a Perspective session:

rosters = system.roster.getRosters()

# Iterate over the rosters, extracting the name and user lists
for name, users in rosters.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.
rosters = system.roster.getRosters()

# Specify the roster with the key (the name of the roster), and iterate over the users.
for user in rosters['myRoster']:

        # output the users
        system.perspective.print(user)
```

Keywords

system roster getRosters, roster.getRosters

system.roster.removeUsers

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

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 the roster. User objects can be created with the system.user.getUser and system.user.addUser functions.

- Returns

  none

- 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, deviceId [, 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 deviceId - 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

  None

- Scope

  Designer, 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 **active Port**, 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

  None

- 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)
```

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, 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.

  Integer poll - Specifies in milliseconds how often to poll the system for a response. If omitted the poll will be 150 milliseconds.

- Returns

  Object - 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, "yyMMddHHmmss")
```

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 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

  None

- 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. Example: "S1F13"

  Boolean reply - Whether or not the SECS message expects a reply message.

  Object 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

  None

- 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

  Object 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
      }
   ]
}
```

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 Tree View comp onent.

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 that can be used in the Tree View component.

- 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

  Int transactionID - The TxID of the response. The TxID from the received request in the payload of the message handler must be specified here.

  Int 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. Example: "S1F14"

  Object 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

  None

- 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

  None

- Returns

  PyTuple - A list of the roles (strings) that are assigned to the current user.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This would run 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

  none

- Returns

  String - The current user name.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This code would run on a startup script and does 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. Optional. Leaving this out will use the project's default profile.

  Integer timeout - Timeout for client-to-gateway communication. (default: 60,000ms)

- Returns

  PyTuple - A list of the roles that this user has, if the user authenticates successfully. Otherwise, returns None.

- Scope

  Gateway, Vision Client

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

  none

- 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 thesystem.security.unlockScreen() function.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.security.lockScreen([obscure])

- Parameters

  boolean obscure - If true(1), 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

  none

- 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 (1), no error will be shown if the switch user function fails. (default: 0)

- Returns

  boolean - false(0) if the switch user operation failed, true (1) 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

  none

- 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. Optional. Leaving this out will use the project's default profile.

  Integer timeout - Timeout for client-to-gateway communication. (default: 60,000ms)

- Returns

  boolean - false(0) if the user failed to authenticate, true(1) if the username/password was a valid combination.

- Scope

  Vision Client

Syntax

system.security.validateUser(username, password, authProfile)

- Parameters

  String username - User name to validate.

  String password - User's password.

  String authProfile - The name of the authentication profile to run against.

- Returns

  boolean - True if valid username/password combination.

- Scope

  Gateway

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 examples associated with this scripting 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, 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:

    ```
    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:

    ```
    system.serial.DATA_BITS_5, system.serial.DATA_BITS_6, system.serial.
    DATA_BITS_7, system.serial.DATA_BITS_8
    ```

    Boolean hardwareFlowControl - Configure hardware flow control. On or off.

    Integer parity - Configure parity. Valid values are defined by the following constants:

    ```
    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:

    ```
    system.serial.STOP_BITS_1, system.serial.STOP_BITS_2
    ```

Note: The serial library was updated in 8.0. Any constants, like HANDSHAKE, that do not have an equivalent value will result in a value of 0.

- Returns

    SerialConfigurator - A list of SerialConfigurator objects that can be used to configure the serial port instead of or in addition to the given keyword arguments.

- 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 examples associated with this scripting function.

Keywords
system serial openSerialPort, serial.openSerialPort

system.serial.port

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

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.

  int numberOfBytes - The number of bytes to read.

  int timeout - Maximum amount of time, in milliseconds, to block before returning. Default is 5000. [optional]

- Returns

  byte[] - A byte[] containing bytes read from the serial port.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no examples associated with this scripting 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 system.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.

  int numberOfBytes - The number of bytes to read.

  int timeout - Maximum amount of time, in milliseconds, to block before returning. Default is 5000. [optional]

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

8.0.15 introduced the following parameter:

  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 examples associated with this scripting 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.

  int 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.

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.

  int timeout - Optional timeout in milliseconds. Default is 5000.

- 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 examples associated with this scripting 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".

  int millis - Approximate length of break signal, in milliseconds.

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

There are no examples associated with this scripting 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, [timeout], [encoding])

- Parameters

  String port - The previously configured serial port to use.

  String toWrite - The String to write.

  The following feature is new in Ignition version **8.0.4**
  Click here to check out the other new features

  Integer timeout - A timeout, in milliseconds. Writes exceeding this period will . Defaults to 5000 [optional]

  The following feature is new in Ignition version **8.0.15**
  Click here to check out the other new features

  String encoding- Optional encoding to decode the string with (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):

- `IS0-8859-1`
- `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 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.

  byte[] toWrite - The byte[] to write.

- Returns

  nothing

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no examples associated with this scripting 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

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.

There are a number of built-in variables maintained by the SFC engine that can be read through the `chart` scope.

Variable Name	Description
chart.instanceId	The string UUID of the running chart instance
chart.startTime	A `java.util.Date` object that indicates when the chart instance started running.
chart.runningTime	An integer representing the number of seconds the chart has been running for.
chart.parent	The chart scope of the enclosing chart (if any). null if this chart was not executed as part of an enclosing step.

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 chart will enter Canceling state.

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

  id -The ID of the chart instance to cancel

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

- Throws

  Will throw a KeyError if the ID does not match any running chart instance.

Code Examples

Code Snippet

```
# The following will attempt 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 will check 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(instanceId)

- Parameters

  String instanceId - 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
instanceId	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 **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.	Dictionary
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 Pausing state.

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

  id - The ID of the chart instance to pause

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

- Throws

  Will throw a KeyError if the ID does not match any running chart instance.

Code Examples
There are no examples associated with this scripting 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 examples associated with this scripting 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.

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

  id - The ID of the chart instance to resume.

- Returns

  Nothing

- Scope

  Gateway, Vision Client, Perspective Session

- Throws

  Will throw a KeyError if the ID does not match any running chart instance.

Code Examples

There are no examples associated with this scripting 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 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 - [Optional] The id for a step inside of a chart. If omitted the function will target a chart scoped variable.

  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 = "SFCMessage", 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 - [Optional] The id for a step inside of a chart. If omitted the function will target a chart scoped variable.

  PyObject variablesMap - 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 a 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(projectName, chartPath, parameters)

- Parameters

  projectName - The name of the project that the chart was created in.

  chartPath - The path to the chart, for example "ChartFolder/ChartName"

  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 will start 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.

  PyDictionary 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. Use getResults() on the results object to get the list of tag dictionaries, or getReturnedSize() to get the number of tags returned by the browse.

- 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.	`# Literally search for "MyTag"` `{"name":"MyTag"}` `# Searches for any names that contain "Tag" in their name` `{"name":"*Tag*"}`
`dataType`	Represents the data type on the tag. Valid values can be found on the Tag Properties page .	`{"dataType":"Int4"}`
`valueSource`	Represents how the node derives its value. Generally only used by nodes with a tagType of "AtomicTag". Valid values can be found on the Value Source description on the Tag Properties page	`{"valueSource":"opc"}`
`tagType`	The type of the node (tag, folder, UDT instance, etc). A list of possible types can be found on the Tag Properties page	`{"tagType":"AtomicTag"}`
`typeId`	Represents the UDT type of the node. If the node is a UDT definition, then the value will be `None`. 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 **UdtInstance.**	`{"typeId":"myUDT", "tagType":"UdtInstance"}`
`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.	`{"quality":"Good"}`
`maxResults`	Limits the amount of results that will be returned by the function.	`{"maxResults":10}`

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`	A fully qualified tag path to the node, including the name of the node.
`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 browse 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 contains the following keys:

Key	Description
`dataType`	The data type of the tag.
`valueSource`	Represents how the tag derives its value.
`value`	The last known qualified value on the tag.

UDT Keys

Both UDT Instances and UDT Definitions add the following key:

Key	Description
`tagType`	Represents the type the UDT is based off of. UDT Definitions will have a value of `None` .

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 call itself to 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.

# Create the function
def browseTags(path, filter):

         # First, browse for anything that can have children (Folders and UDTs, generally)
        results = system.tag.browse(path)
        for branch in results.getResults():
            if branch['hasChildren']:
                # If something has a child node, then call this function again so we can
search deeper.
                # Include the filter, so newer instances of this call will have the same
filter.
                browseTags(branch['fullPath'], filter)


    # Call this function again at the current path, but apply the filter.
        results = system.tag.browse(path, filter)
        for result in results.getResults():

                # Here's where you'd want to do something more useful. For now we print.
                print result['fullPath']

# Call the function. Replace the filter with your own search criteria.
browseTags('[default]', {'tagType':"UdtInstance"})
```

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.

  String[] nameFilters - A list of name filters to be applied to the result set. [optional]

  Integer maxSize - The maximum size of the result set. [optional]

  Object 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. [optional]

- Returns

  Results - A Results object which contains a list of tag dictionaries, one for each tag found during the browse. Use getResults() on the results object to get the list of tag dictionaries, or getReturnedSize() to get the number of tags returned by the browse.

- 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.
```

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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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 existing tag(s), not the tag(s) themselves.

  Object 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. 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.
  - 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.

- 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 a base path. Sometime useful, but we could hardcode the paths
#   in the getConfiguration() and configure() calls
parentPath = "[NewProvider]SomeFolder"

# Get the current configurations recursivly, 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

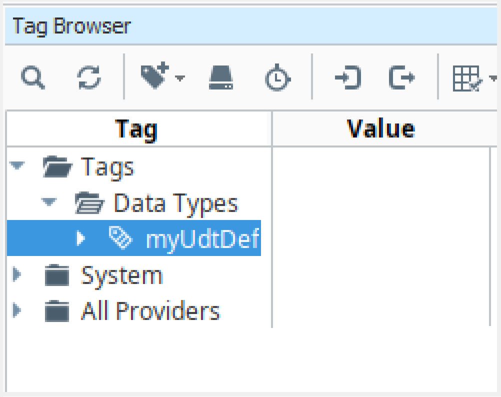
```
# This example will add a new UDT Instance. It can be further expanded to modify more
# properties on the tag. Additionally, this example can be used to edit an existing tag
# by setting the baseTagPath to a tag that already exists, and by modifying the collision
policy.

# 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)
```

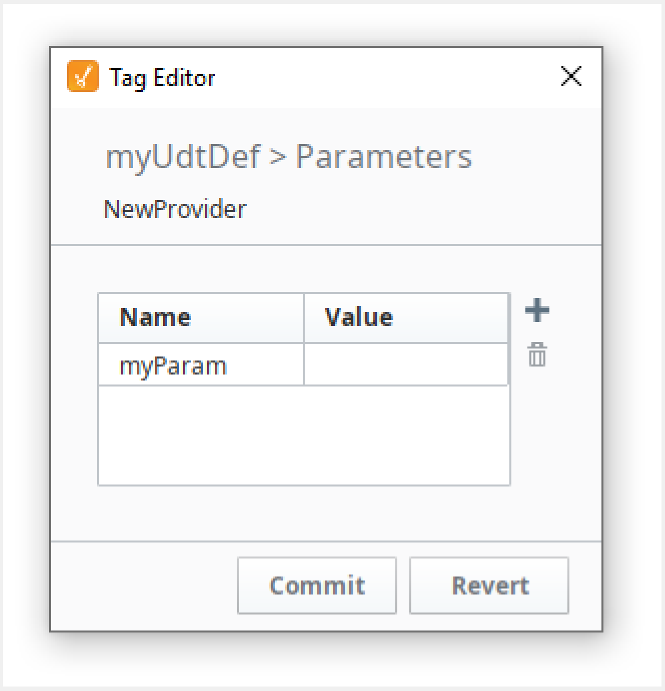
Python - Adding Folders in other Folders

```
# Folders are nodes with a 'tagType' set to 'Folder'
# Each folder can contain a 'tags' value, which containers other tags and folders.

Tags={'tagType': 'Folder',
                'name': 'NewFolderName',
                'tags' : [
                                        {
                                        'name': 'anotherfolder',
                                        'tagType': 'Folder',
                                        'tags': [{}]                        # There aren't
any objects defined here, so this will just be an empty
folder.
                                        }
                                ]
                }

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`").





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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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 - [optional] 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 Abort.

- Returns

  List - A List of QualityCode objects, one for each tag in the list, that is representative of the result of the operation.

- 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.deleteTags

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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, e.g. "Good", "Bad_NotFound", etc. The quality codes have a built-in isNotGood() method that can be used to determine if any deletions failed.

- 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

    All

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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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 - [optional] Set to True to export all Tags under each Tag path, including Tags in child folders. Defaults to True

  String exportType - The type of file that will be exported. Set to "json" or "xml". Defaults to "json".

- Returns

  None

- 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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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.

Note:

The configurations returned by this function can only contain properties that are **not** using their default values. Thus, if a property on a tag has not been modified from its default value, then it will be included in the returned list.

Should you need to read the value of a tag property, regardless of whether it's using the default value or not, use system.tag.readBlocking instead:

```
system.tag.readBlocking(["[default]path/to/tag.engUnit"])
```

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 name 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 name property in a similar manner
#    to accessing a key in a Python Dictionary.

print config[0]['name']
```

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
# recursivly
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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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 - [optional] 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.

- Returns

  List - A List of QualityCode objects, one for each tag in the list, that is representative of the result of the operation.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no examples associated with this scripting 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

  none

- Returns

  boolean - True (1) if overlays are currently enabled.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples
There are no examples associated with this scripting function.

Keywords
system tag isOverlaysEnabled, tag.isOverlaysEnabled

system.tag.move

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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 - [optional] 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.

- Returns

  List - A List of QualityCode objects, one for each tag in the list, that is representative of the result of the operation.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Rename a Folder

```
# This function will move 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 will move 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.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

  PySequence 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.

  PySequence 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.

  Date endDate - The end of the calculation window. If omitted, and range is not used, uses the current time.

  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.

  Integer rangeMinutes - Same as rangeHours, but in minutes.

  PySequence 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.

  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". For more information see Seeded Values. [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"

  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"

  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".

- 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

  PySequence 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 2-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 on.
- "SimpleAverage" - returns the simple mathematical average of the values - ((V1+V2+...+Vn)/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)

- Parameters

  PySequence 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.

  Date endDate - The latest value to retrieve. If omitted, current time is used.

  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. Numbers larger than that will determine the number of points that should be returned. -1 is the default.

  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". Default is "Average" (time-weighted).

  String returnFormat - Use "Wide" to have a column per tag queried, or "Tall" to have a fixed-column format. Default is "Wide".

  PySequence 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.

  Integer intervalHours - Allows you to specify the window interval in terms of hours, as opposed to using a specific return size.

  Integer intervalMinutes - Same as intervalHours, but in minutes. Can be used on its own, or in conjunction with intervalHours.

  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.

  Integer rangeMinutes - Same as rangeHours, but in minutes.

  PySequence aggregationModes - A one-to-one list with paths specifying an aggregation mode per column.

  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. For more information see Seeded Values. [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.

  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.

  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.

  Integer timeout - Timeout in milliseconds for Client Scope. This property is ignored in the Gateway Scope. [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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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

Description

Asynchronously reads the value of the Tags at the given paths. Meaning, execution of the calling script will continue without waiting for this function to finish. This useful in cases where the rest of the script should continue without waiting for the results from system.tag.readAsync.

Instead of returning the tag read results to the calling script, the results are processed by a Python callback function.

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

  None

- 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

        # Replace this part of the function with something more useful, such as
        #    a tag or DB write.
        print 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

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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

Description

Reads the value of the Tags at the given paths. Will "block" until the read operation is complete or times out. Meaning, execution of the calling script will pause until this function finishes. This useful in cases where the tag read must complete before further lines in the same script should execute.

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. Each qualified value will have three sub members: **value**, **quality**, and **timestamp**.

- 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.rename

The following feature is new in Ignition version **8.0.2**
Click here to check out the other new features

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

Description

Renames a single tag or folder.

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.rename(tag, newName, [collisionPolicy])

- Parameters

  String tag - A path to the tag or folder to rename.

  String newName - The new name for the tag or folder.

  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 Abort if not specified. [optional]

- Returns

  QualityCode - A QualityCode object that contains the results of the rename operation.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Rename a Folder

```
old = "[default]folder/tag"
new = "noun_1"                # Note that the new name should not include the full path or tag provider. Just the name suffices.

system.tag.rename(old, new)
```

Keywords

system tag rename, tag.rename

system.tag.requestGroupExecution

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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 associated with 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 (1) to turn on tag overlays, false (0) to turn them off.

- Returns

  nothing

- Scope

  Client

Code Examples

There are no examples associated with this scripting function.

Keywords

system tag setOverlaysEnabled, tag.setOverlaysEnabled

system.tag.writeAsync

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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

Description

Asynchronously writes values to Tags at specified paths. The script will not wait for the write operation to complete before moving on, but you can provide a callback function to run further code after the write has finished.

For a blocking tag write operation, see system.tag.writeBlocking.

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. A tag property can be included in the path. 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 function that will be invoked after the write operation is complete. The function must take a single argument: a List of QualityCode objects corresponding to the results of the write operation. [optional]

- Returns

  None

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Write to Multiple Tags

```
paths = ["[default]Tag1", "[default]Folder/Tag2"]
values = [10, 10]
system.tag.writeAsync(paths, values)
```

Code Snippet - Using a Callback Function

```
paths = ["[default]Tag1", "[default]Folder/Tag2"]
values = [10, 10]

def myCallback(asyncReturn):
        for result in asyncReturn:
                # Do something if a bad qualified value was returned
                if not result.good:
                        print result

system.tag.writeAsync(paths, values, myCallback)
```

Keywords

system tag writeAsync, tag.writeAsync

system.tag.writeBlocking

The following feature is new in Ignition version **8.0.0**
Click here to check out the other new features

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

Description

Writes values to Tags at the given paths. This function will "block" until the write operation is complete or times out. Meaning, execution of the calling script will pause until this function finishes. This useful in cases where the tag write must complete before further lines in the same script should execute.

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. Each quality code holds the result of the write operation for that Tag.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Write to Multiple Tags

```
# Create a list with the tag paths to be updated.
paths = ["[default]Folder/Tag_A","[default]Folder/Tag_B"]

# Create a list with the update values, one value for each path.
values = [1,2]

# Execute the write operation.
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

    None

- 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

  None

- 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 to retrieve phone numbers for

- 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 to retrieve phone numbers for

- 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 a 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 a SMS message.
# Fetch the Twilio account name
# getAccounts() returns a list, so the "[0]" operator is refering 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

The following feature is new in Ignition version **8.0.11**
Click here to check out the other new features

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
Adds a holiday.

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.getInfos()
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.getInfos()
        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

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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 the AcmeWest usersource.
system.user.addUser("AcmeWest", userToGet)
```

Keywords

system user addUser, user.addUser

system.user.createScheduleAdjustment

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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.getInfos()
        print "Infos are:"
        printResponse(infos)

"""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 edit. Can be a BasicScheduleModel or CompositeScheduleModel obje ct (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

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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.isRepeatAnually()

"""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

  None

- 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

  none

- 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

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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 reposonses
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", "Operator"]
user.addRoles(roles)

# and we can remove a role
user.removeRole("Operator")

# 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.getInfos()
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

  none

- 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 getScheduledNames, user.getScheduledNames

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

  none

- 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.getSchedules

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')

...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

You can access most of the basic user properties via a call to "get" which returns a null value if the requested item is not present. For example:

user.get("firstname")

...will return that user's first name, or a null value if a name hasn't been set. The following list represents 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 the schedule for.

  Date/Long 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

  Bool - 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. Name is 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.getInfos()
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. Name is 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

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

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

The following feature is new in Ignition version **8.0.13**
Click here to check out the other new features

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], [actionTarget], [actionValue], [auditProfile], [actor], [actorHost], [originatingSystem], [eventTimestamp], [originatingContext], [statusCode])

- Parameters

  String action - What happened. [Optional] Default is null.

  String actionTarget - What the action happened to. [Optional] Default is null.

  String actionValue - The value of the action. Default is no value. [optional]

  String auditProfile - Where the audit record should be stored. [Optional] Defaults to the project's audit profile (if specified), or the gateway audit profile if calling in the gateway or perspective scope.

  String actor - Who made the change. [Optional] Will be populated automatically if omitted, assuming there is a known user;

  String actorHost - The hostname of whoever made the change. [Optional] Will be populated automatically if omitted.

  String, List 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:${gatewayName}:\project:${projectName}`. So if you provided `[“component”, “Joe’sButton”, “field”, “value”]`, you would get a record with `originatingSystem:sys:${gatewayName}:\project:${projectName}:\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.

  Date eventTimestamp - When the event happened. [Optional] Will be set to the current time if omitted.

  Integer originatingContext - What scope the event originated from: 1 means Gateway, 2 means Designer, 4 means Client. [Optional] Will be set automatically if omitted.

  Integer statusCode - A quality code to attach to the object. Defaults to 0, indicating no special meaning.

- 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 vary least you can always provide just the action you wish to record, allowing 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

  Vision Client

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

  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 function flushes the resolver cache.
system.util.execute(["ipconfig", "/flushdns"])
```

Code Snippet

```
# This code starts the Firefox browser and opens the Google home page.
# Although system.util.execute() is also Perspective Session scoped, the following code
snippet
# will not work in a Perspective Session due to limitations of launching programs from a web
environment.
system.util.execute(['C:\\Program Files\\Mozilla Firefox\\firefox.exe', 'https://www.google.
com'])
```

Code Snippet

```
# This code runs the Notepad program and opens the Ignition license text file.
# Although system.util.execute() is also Perspective Session scoped, the following code
snippet
# will not work in a Perspective Session due to limitations of launching programs from a web
environment.
system.util.execute(['notepad.exe', 'c:\\program files\\inductive
automation\\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 (1), the shutdown-intercept script will be skipped. Default is false (0). [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

  none

- 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

  none

- 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

  none

- Returns

  String - A special code representing the client's session in a unique way.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This code would print 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 session'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

  none

- Returns

  int - The current connection mode for the client.

- Scope

  Vision Client

Code Examples

```
# This code will set 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

  none

- Returns

  int - The current connect timeout, in milliseconds. Default is 10,000 (ten seconds)

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This code would print 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

  none

- Returns

  String - The edition of the Vision module that is running the client.

- Scope

  Vision Client

Code Examples

```
# This code will write 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 that the client is currently communicating with.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.util.getGatewayAddress()

- Parameters

  none

- Returns

  String - the address of the Gateway that the client is communicating with.

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This code would open 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 - Optional. The maximum time in milliseconds to attempt to initially contact a Gateway.

  Integer socketTimeoutMillis - Optional. The maximum time in milliseconds to wait for a response from a Gateway after initial connection has been established.

  The following feature is new in Ignition version **8.0.11**
  Click here to check out the other new features

  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.

- 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.

Caution: The contents of the system.util.getGlobals() dictionary is cleared out every time the originating project is saved.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.util.getGlobals()

- Parameters

  none

- Returns

  PyStringMap - 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.

⚠ 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

  none

- Returns

  long - 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.util.getLocale

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

  none

- Returns

  String

- 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 Tool s > 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 would log 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.info("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

  none

- Returns

  String - The name of the currently running project.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code would display the name of the currently running project
system.gui.messageBox("You are running project: %s" % system.util.getProjectName())
```

Keywords

system util getProjectName, util.getProjectName

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 would store 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%smyfile.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

  none

- Returns

  int - 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)

Note that 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.

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 would get the entire table of sessions and put 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 would count 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 would return 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 would return 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 cont ains 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

  none

- Returns

  int - 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 will take the integer representing the system flags, convert
it to bits, and place it in a list and then print it out.
# The second part of the script will take the list of bits, and place 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

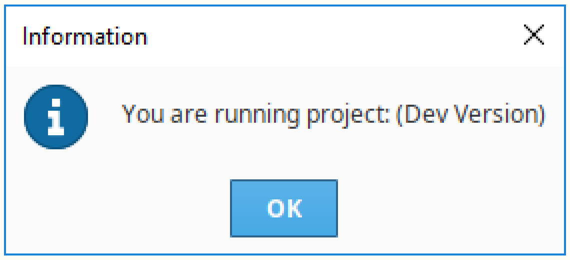
The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

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

Description

Returns the Ignition version number that is currently being run.

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:


Information

You are running project: (Dev Version)

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.util.getVersion()

- Parameters

  none

- 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 would display the name of the currently running Ignition version number
system.gui.messageBox("You are running project: %s" % system.util.getVersion())
```

Code Snippet

```
# This code would display 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

This is an advanced scripting function. 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:

This function should not be used to asynchronously interacts with the GUI in Vision. This means interacting with window navigation, setting and getting component properties, showing error/message popups, and really any other methods that can interact with components and windows. If you need to do something with the GUI in Vision with this function, this must be achieved through a subsequent call to system.util.invokeLater.

By contrast, this function is safe to use in the gateway, which also means calls from Perspective are safe.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.util.invokeAsynchronous(function, [args], [kwargs], [description])

- Parameters

  PyObject function - A Python function object that will get invoked with no arguments in a separate thread.

  Iterable args - A list or tuple of Python objects that will be provided to the called function as arguments. Equivalent to the * operator. [optional]. Added in 8.0.15.

  Dictionary 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]. Added in 8.0.15.

  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]. Added in 8.0.15.

- 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.invokeLater(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

This is an advanced scripting function. 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

  PyObject function - A Python function object that will be invoked later, on the GUI, or event-dispatch, thread with no arguments.

  int 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 dict. 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

  PyObject - 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
Array	PyList
Integer	Integer
null	None
true/false (boolean)	True/False (boolean)
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 datatype 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 dict and converts into a json string.

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.util.jsonEncode(pyObj, [indentFactor])

- Parameters

  PyObject pyObj - The Python object to encode into JSON such as a Python list or dictionary.

  int indentFactor - Optional parameter. The number of spaces to add to each level of indentation for prettyprinting.

- 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
List or Tuple	Array
Integer	Integer
None	null
True/False (boolean)	true/false (boolean)
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

  Vision Client

The following feature is new in Ignition version **8.0.13**
Click here to check out the other new features

As of version 8.0.13, the scope of this function was expanded to include the Gateway and Perspective Sessions.

Code Examples

Code Snippet

```
# This code adds or updates a translation into French
# for the world 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 byte[].

Client Permission Restrictions

This scripting function has no Client Permission restrictions.

Syntax

system.util.playSoundClip(wavBytes [, volume] [, wait])

- Parameters

  byte[] wavBytes - A byte list of a wav file.

  double 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

  Vision Client

Syntax

system.util.playSoundClip(wavFile [, volume] [, wait])

- Parameters

  String wavFile - A filepath or URL that represents a wav file

  double 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

  Vision Client

Code Examples

Code Snippet

```
# This code would play a sound clip at full volume that was located on the current
# host's filesystem. It will not return until the clip in 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.

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

In version 8.0.3 this function was added to the gateway scope, using the syntax listed below:

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.

  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

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] (Note that the project must have an Audit Profile configured in Project Properties. Otherwise this parameter is mandatory)

  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

  Perspective Session, 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
# (since the startDate parameter is omitted), 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

As of version 8.0.8, 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**. [optional]

  The following feature is new in Ignition version **8.0.8**
  Click here to check out the other new features

  As of 8.0.8 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.

  PyDictionary 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]

  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.retarget("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.retarget("TankControl", "10.30.2.34:8043")
```

Code Snippet

```
# This code would be put in a button in the target that was retargetted 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.retarget(project, gateway)
```

Keywords

system util retarget, util.retarget

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.

Note that 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.

  PyDictionary payload - Optional. A PyDictionary which will get passed to the message handler. Use "payload" in the message handler to access dictionary variables.

  String scope- Optional. Limits the scope of the message delivery to "C" (clients), "G" (Gateway), "CG" for clients and the Gateway, or "S" Session (as of version 8.0.7). Any combination of C, G, and S are available.

  Prior to version 8.0.7: Defaults to "C" if the user name, role or host name parameters are set, and to "CG" if none of these parameters are set.
  As of version 8.0.8: Defaults to "CS" if the user name, role, or host name parameters are set, and to "CGS" if none of these parameters are set.

  String clientSessionId - Optional. Limits the message delivery to a client with the specified session ID.

  String user - Optional. Limits the message delivery to clients where the specified user has logged in.

  String hasRole - Optional. Limits the message delivery to any client where the logged in user has the specified user role.

  String hostName - Optional. Limits the message delivery to the client that has the specified network host name.

  List remoteServers - Optional. 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.

- 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

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

As of 8.0.7 system.util.sendMessage can send messages to both Perspective browser and mobile sessions.

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.

  PyDictionary payload - Optional. A PyDictionary which will get passed to the message handler. Use "payload" in the message handler to access dictionary variables.

  String hostName - Optional. Limits the message delivery to the client that has the specified network host name.

  String remoteServer - Optional. 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.

  String timeoutSec - Optional. The number of seconds before the sendRequest call times out.

- 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.

  PyDictionary payload - Optional. A PyDictionary which will get passed to the message handler. Use "payload" in the message handler to access dictionary variables.

  String hostName - Optional. Limits the message delivery to the client that has the specified network host name.

  String remoteServer - Optional. 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.

  String timeoutSec - Optional. The number of seconds before the sendRequest call times out.

  PyFunction onSuccess- Optional. 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.

  PyFunction onError- Optional. Should take one argument, which will be the exception encountered. Callback functions will be executed on the GUI thread, similar to system.util.invokeLater .

- 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 will call the message handler 'test', and will return 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 session. 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

  int 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, would check the current
username and set 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

  int connectTimeout - The new connect timeout, specified in milliseconds.

- Returns

  nothing

- Scope

  Vision Client

Code Examples

Code Snippet

```
# This code would set 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.

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

  None

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
system.util.setLoggingLevel("Reporting", "debug") # This would set the logger called
Reporting to the debug level.
```

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

  int readTimeout - The new read timeout, specified in milliseconds.

- Returns

  nothing

- Scope

  Vision Client

Code Examples

```
# This script will set 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

  None

- Returns

  String The dump of the current running JVM.

- Scope

  Gateway, Vision Client, Perspective Session

Code Examples

```
# This script will take a thread dump of the current JVM, and write 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

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

  Vision Client

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

As of version 8.0.8, the scope of this function was expanded to include the Gateway and Perspective.

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