SkanIt™ Automation Interface Client API v3.0

Table of contents

Namespace Thermo.MIP.Automation.Client

Classes

Client

SiLA 2 client for connecting to SiLA 2 automation server and using the functionality of its features.

Class Client

SiLA 2 client for connecting to SiLA 2 automation server and using the functionality of its features.

Inheritance

System.Object
Client

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: Thermo.MIP.Automation.Client

Assembly: Thermo.MIP.Automation.Client.dll

Syntax

```
public class Client
```

Constructors

Client()

Constructor.

Declaration

```
public Client()
```

Properties

IsChannelOpen

True if there is currently channel open, otherwise false. Note that this is not a guarantee that any server exists and is listening.

Declaration

```
public bool IsChannelOpen { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Abort()

Abort instrument command or session execution.

Declaration

```
public void Abort()
```

CloseChannel()

Close channel to SiLA server

Close channel to SiLA server.

Declaration

```
public void CloseChannel()
```

Connect(String, Int32)

Connect instrument.

Declaration

```
public void Connect(string serialNumber, int statusEventInterval)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	Instrument serial number.
System.Int32	statusEventInterval	Interval of status events in milliseconds.

Disconnect()

Disconnect instrument.

Declaration

```
public void Disconnect()
```

ExecuteSession(String, String)

Execute session.

Declaration

```
public void ExecuteSession(string filePath, string executedFilePath = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filePath	Session file path.
System.String	executedFilePath	File path where to save the executed session. If the file already exists, running number will be inserted to executed session file name. If empty, system uses the original session file path with running number inserted to session file name.

Remarks

Path can be absolute or relative. If relative, it is relative to the Thermo.MIP.Automation.Server.exe. Ensure write access to the given location.

GetAtmosphere()

Get oxygen and carbon dioxide levels.

Declaration

```
public Dictionary<string, double> GetAtmosphere()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Double>	Dictionary with keys 'OxygenLevel' and 'CarbonDioxideLevel'. Values are zero if atmospheric control turned off (see remark).

Remarks

Atmospheric control is only supported by Varioskan LUX. Atmospheric control is initially turned off and has to be turned on with either SetAtmosphereO2(), SetAtmosphereCO2() or SetAtmosphere() otherwise GetAtmosphere() will return 0 and atmosphere readings will not be included in StatusData.

GetErrors()

Get instrument errors.

Declaration

```
public void GetErrors()
```

Remarks

The GetErrors command is not supported by Fluoroskan, Fluoroskan FL and Luminoskan instruments.

GetFeatureDefinition(String)

Get the Feature Definition of an implemented Feature by its fully qualified Feature Identifier. This command has no preconditions and no further dependencies and can be called at any time.

Declaration

```
public string GetFeatureDefinition(string featureIdentifier)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	featureIdentifier	

Returns

TYPE	DESCRIPTION
System.String	The Feature definition in XML format (according to the Feature Definition Schema).

GetImplementedFeatures()

Returns a list of fully qualified Feature identifiers of all implemented Features of this SiLA server. This list remains the same throughout the lifetime of the SiLA server.

Declaration

```
public IList<string> GetImplementedFeatures()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IList<System.String>	List of feature identifiers.

GetInstrumentInfo()

Gets information about the connected instrument.

Declaration

```
public InstrumentInfo GetInstrumentInfo()
```

Returns

TYPE	DESCRIPTION
InstrumentInfo	Instrument info.

GetInstrumentStatus()

Gets the status of the connected instrument.

Declaration

```
public string GetInstrumentStatus()
```

Returns

TYPE	DESCRIPTION
System.String	Instrument status.

Remarks

This can be used to query the overall status of the server. Whether an instrument is connected or not and if connected in which state the instrument currently is.

GetReport(String)

Get a report from the last session execution. Report contains all measurement and result steps.

Declaration

```
public void GetReport(string reportFilePath)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	reportFilePath	Report target file path.

Remarks

The extension of the target file given in 'reportFilePath' defines the format of the the report. Possible extensions are *.xml, *.pdf, *.xlsx, *.txt

GetResultFormat()

Get format for measurement results.

Declaration

```
public string GetResultFormat()
```

Returns

TYPE	DESCRIPTION
System.String	Result format.

Remarks

Not yet supported. Currently 'Skanit_7.0' is always returned.

GetServerDescription()

Description of the SiLA server. Include the use and purpose of this SiLA server.

Declaration

```
public string GetServerDescription()
```

Returns

TYPE	DESCRIPTION
System.String	Server description.

GetServerName()

Human readable name of the SiLA server. The name can be set using the 'SetServerName' method.

Declaration

```
public string GetServerName()
```

Returns

TYPE	DESCRIPTION
System.String	Server name.

GetServerType()

The type of this server. It, could be, e.g., in the case of a SiLA Device the model name. It is specified by the implementer of the SiLA server and MAY not be unique.

Declaration

```
public string GetServerType()
```

Returns

TYPE	DESCRIPTION
System.String	Server type.

GetServerUuid()

Globally unique identifier that identifies a SiLA server. The server UUID is generated once and remain the same for all times.

Declaration

```
public Guid GetServerUuid()
```

Returns

TYPE	DESCRIPTION
System.Guid	Server UUID.

GetServerVendorUri()

Returns the URI to the website of the vendor or the website of the product of this SiLA server.

Declaration

```
public Uri GetServerVendorUri()
```

Returns

TYPE	DESCRIPTION
System.Uri	Server vendor URI.

GetServerVersion()

Returns the version of the SiLA server.

Declaration

```
public Version GetServerVersion()
```

Returns

TYPE	DESCRIPTION
System.Version	Server version.

GetTemperature()

Gets the temperature information from the instrument.

Declaration

```
public Dictionary<string, double> GetTemperature()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.String, System.Double>	Current and target temperatures in °C.

Remarks

Temperature not available in all instruments. Some instruments report temperatures separately for incubator and cuvette.

OpenChannel(IPAddress, Int32)

Open channel to the SiLA server.

Declaration

```
public bool OpenChannel(IPAddress ip, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
IPAddress	ip	IP address.
System.Int32	port	Port number.

Returns

TYPE	DESCRIPTION
System.Boolean	True if opening the channel was successful, otherwise false.

OpenChannel(String, Int32)

Open channel to the SiLA server.

Declaration

```
public bool OpenChannel(string ip, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	ip	IP address
System.Int32	port	Port number

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Boolean	True if opening the channel was successful, otherwise false.

PlateIn()

Move plate tray in.

Declaration

```
public void PlateIn()
```

Remarks

Instrument will enter Busy-state and very shortly afterwards go to IDLE state even if movement is still ongoing. Commands issued during the time of movement will be queued up.

PlateOut()

Move plate tray out.

Declaration

```
public void PlateOut()
```

Remarks

Instrument will enter Busy-state and very shortly afterwards go to IDLE state even if movement is still ongoing. Commands issued during the time of movement will be queued up.

SetAtmosphere(Double, Double)

Set oxygen and carbon dioxide levels.

Declaration

```
public void SetAtmosphere(double o2, double co2)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	o2	Target oxygen percentage level. Value will be rounded to one decimal place. Midpoints rounded away from zero.
System.Double	co2	Target carbon dioxide percentage level. Value will be rounded to one decimal place. Midpoints rounded away from zero.

Remarks

Atmospheric control is only supported by Varioskan LUX.

SetAtmosphereCO2(Double)

Set carbon dioxide level.

Declaration

```
public void SetAtmosphereCO2(double co2)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	co2	Target carbon dioxide percentage level. Value will be rounded to one decimal place. Midpoints rounded away from zero.

Remarks

Atmospheric control is only supported by Varioskan LUX.

SetAtmosphereO2(Double)

Set oxygen level.

Declaration

```
public void SetAtmosphereO2(double o2)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	o2	Target oxygen percentage level. Value will be rounded to one decimal place. Midpoints rounded away from zero.

Remarks

Atmospheric control is only supported by Varioskan LUX.

SetResultFormat(String)

Set format for measurement results.

Declaration

```
public void SetResultFormat(string format)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	format	Measurement results format.

Remarks

Setting result format is not yet supported, support planned for a future release.

SetServerName(String)

Sets a human readable name to the SiLA server Name Property. This command has no preconditions and no further dependencies and can be called at any time.

Declaration

```
public void SetServerName(string serverName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serverName	Server name.

SetTemperature(Double)

Set target temperature.

Declaration

```
public void SetTemperature(double temperature)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	temperature	Temperature in °C.

SubscribeInstrumentEvents()

Subscribe instrument event messages from the SiLA server. InstrumentEvent will be invoked when a new event message is received.

Declaration

```
public void SubscribeInstrumentEvents()
```

SubscribeInstrumentStatus()

Subscribe instrument status messages from the SiLA server. InstrumentStatusChanged will be invoked when a new status message is received.

Declaration

```
public void SubscribeInstrumentStatus()
```

UnsubscribeInstrumentEvents()

Unsubscribe instrument event messages from the SiLA server.

Declaration

```
public void UnsubscribeInstrumentEvents()
```

UnsubscribeInstrumentStatus()

Unsubscribe instrument status messages from the SiLA server.

Declaration

```
public void UnsubscribeInstrumentStatus()
```

Events

GrpcError

Occurs when gRPC communication layer reports an error.

Declaration

```
public event EventHandler<RpcException> GrpcError
```

Event Type

TYPE	DESCRIPTION
System.EventHandler<Grpc.Core.RpcException>	

InstrumentEvent

Occurs when an event is received from the instrument. See SubscribeInstrumentEvents() and UnsubscribeInstrumentEvents() for subscribing/unsubscribing event messages from the SiLA server.

Declaration

```
public event EventHandler<InstrumentEventArgs> InstrumentEvent
```

Event Type

TYPE	DESCRIPTION
System.EventHandler<InstrumentEventArgs>	

InstrumentStatusChanged

Occurs when instrument status is changed. See SubscribeInstrumentStatus() and UnsubscribeInstrumentStatus() for subscribing/unsubscribing status messages from the SiLA server.

Declaration

```
public event EventHandler<InstrumentStatusEventArgs> InstrumentStatusChanged
```

Event Type

TYPE	DESCRIPTION
System.EventHandler<InstrumentStatusEventArgs>	

SilaError

Occurs when SiLA error is reported via gRPC communication layer.

Declaration

```
public event EventHandler<SiLAError> SilaError
```

Event Type

TYPE	DESCRIPTION
System.EventHandler<SiLAError>	

Namespace Thermo.MIP.Automation.Common

Classes

InstrumentEventArgs

Contains the event data for instrument events.

InstrumentInfo

Contains basic information about instrument.

InstrumentStatusEventArgs

Contains the event data for instrument status events.

SimulatorInfo

Contains basic information about simulators.

Enums

InstrumentEventType

Instrument event type.

InstrumentStatus

Instrument status.

PlatePosition

Position of the instrument's plate tray.

Class InstrumentEventArgs

Contains the event data for instrument events.

Inheritance

System.Object

InstrumentEventArgs

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public class InstrumentEventArgs : EventArgs
```

Constructors

InstrumentEventArgs(InstrumentEventType, String)

Constructor.

Declaration

```
public InstrumentEventArgs(InstrumentEventType eventType, string data)
```

Parameters

TYPE	NAME	DESCRIPTION
InstrumentEventType	eventType	Event type.
System.String	data	Event data.

Properties

Data

Event data.

Declaration

```
public string Data { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

EventType

Event type.

Declaration

```
public InstrumentEventType EventType { get; }
```

Property Value

TYPE	DESCRIPTION
InstrumentEventType	

Enum InstrumentEventType

Instrument event type.

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public enum InstrumentEventType : int
```

Fields

NAME	DESCRIPTION
Error	Error event.
ExecutionStatus	Session execution status.
InstrumentErrors	Instrument errors.
Results	Measurement results.
RunLog	Measurement run log.
Status	Instrument status.

Class InstrumentInfo

Contains basic information about instrument.

Inheritance

System.Object
InstrumentInfo

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public class InstrumentInfo : object
```

Constructors

InstrumentInfo(String, String, String, Version)

Constructor.

Declaration

```
public InstrumentInfo(string instrumentName, string instrumentType, string serialNumber, Version
firmwareVersion)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	instrumentName	Instrument name assigned by the user.
System.String	instrumentType	Instrument type.
System.String	serialNumber	Serial number.
Version	firmwareVersion	Firmware version.

Properties

FirmwareVersion

Firmware version.

Declaration

```
public Version FirmwareVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	

InstrumentName

Instrument name assigned by the user.

Declaration

```
public string InstrumentName { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

InstrumentType

Instrument type.

Declaration

```
public string InstrumentType { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

SerialNumber

Serial number.

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Enum InstrumentStatus

Instrument status.

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public enum InstrumentStatus : int
```

Fields

NAME	DESCRIPTION
Aborting	Instrument is aborting execution.
Busy	Instrument is busy.
Connected	Instrument is connected.
Connecting	Instrument is connecting.
Disconnected	Instrument is disconnected.
Disconnecting	Instrument is disconnecting.
Error	Instrument is in error state.
Executing	Instrument is executing.
UserAction	Instrument is waiting for user action to complete.

Class InstrumentStatusEventArgs

Contains the event data for instrument status events.

Inheritance

System.Object

InstrumentStatusEventArgs

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public class InstrumentStatusEventArgs : EventArgs
```

Constructors

InstrumentStatusEventArgs(InstrumentStatus, InstrumentStatus)

Constructor.

Declaration

```
public InstrumentStatusEventArgs(InstrumentStatus newStatus, InstrumentStatus oldStatus)
```

Parameters

TYPE	NAME	DESCRIPTION
InstrumentStatus	newStatus	New instrument status.
InstrumentStatus	oldStatus	Old instrument status.

Properties

NewStatus

New instrument status.

Declaration

```
public InstrumentStatus NewStatus { get; }
```

Property Value

TYPE	DESCRIPTION
InstrumentStatus	

OldStatus

Old instrument status.

Declaration

```
public InstrumentStatus OldStatus { get; }
```

Property Value

TYPE	DESCRIPTION
InstrumentStatus	

Enum PlatePosition

Position of the instrument's plate tray.

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public enum PlatePosition : int
```

Remarks

Most instruments do not support querying the plate position. To ensure the plate tray is driven out, issue PlateOut command and wait for the time it take to move out.

Fields

NAME	DESCRIPTION
In	The plate is in.
None	Plate position cannot be determined.
Out	The plate is out.

Class SimulatorInfo

Contains basic information about simulators.

Inheritance

System.Object
SimulatorInfo

Namespace: Thermo.MIP.Automation.Common

Assembly: Thermo.MIP.Automation.Common.dll

Syntax

```
public class SimulatorInfo : object
```

Fields

Simulators

Available simulators: Varioskan LUX (SIMULATOR_3020), Multiskan FC (SIMULATOR_357_T), Multiskan GO (SIMULATOR_1510_C), Multiskan Sky (SIMULATOR_1530_C) and Multiskan SkyHigh (SIMULATOR_1550_C).

Declaration

```
public static readonly SimulatorInfo[] Simulators
```

Field Value

TYPE	DESCRIPTION
SimulatorInfo[]	

Properties

SerialNumber

Serial number assigned to the simulator.

Declaration

```
public string SerialNumber { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Type

Type of the instrument the simulator is simulating.

Declaration

```
public string Type { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

GetIsSimulator(String)

Checks if the provided serial number belongs to a simulator.

Declaration

```
public static bool GetIsSimulator(string serialNumber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	serialNumber	Serial number to be checked.

Returns

TYPE	DESCRIPTION
System.Boolean	True if provided serial number matches with one of the simulator serial numbers, otherwise false.