MTConnect .NET
Software Development Kit (SDK)

Version 2.1.7
March 7, 2011

Table of Contents

Revision History 3
Introduction 4
Architecture 4
SDK Solution and Projects 5
MTConnectAgent Solution 5
MTConnectAgentCore Project 5
MTConnectAgentCoreWindows Project 5
MTConnectAgentCoreWindowsService Project 5
MTConnectAgentCoreWindowsServiceSetup Project 5
Implementation Scenarios 6
Independent Adapter and Independent Agent 6
Combined Adapter and Agent 6
Embedded Agent 7
How to Guide 8
Windows Application Version 8
How do I install the Windows Application Version? 8
How do I run the Windows Application Version? 8
Windows Service Version 9
How do I install the Windows Service Version? 9
How to Run 9
How to Stop 9
How to Uninstall 9
Agent Interaction 10
How do I send an HTTP Request to the Core Agent? 10
How do I send Data to the Core via HTTP? 10
How do I configure the Core Agent? 10
How can I view data stored by the Agent? 10
Development 11
How can I Use the DLL in a Visual Studio 2005 C# Project? 11
Start Function 12
Stop Function 12
StoreSample Function 13
StoreEvent Function 14
StoreCondition Function 15
HTTP Interface 16
debug 17
storeSample 17
storeEvent 18
storeCondition 19
version 19
Agent Configuration 20
SDK Package Structure 21
General 21

Core Agent 21
Windows Executable to Run Core Agent 21
Windows Service to Run Core Agent 21
Create Core Agent Windows Service 22
Known Issues 23
Comments or Questions 23

Revision History

Date	Description	Author	Version
4/30/2008	Initial Draft	Georgia Tech	Prerelease
5/1/2008	Changes and Recommendations from Paul Warndorf.	Georgia Tech	0.1
5/19/2008	Updated images. Added table of contents, terminology, and revision history. Various edits throughput.	Georgia Tech	0.2
6/3/2008	Updated images.	Georgia Tech	0.21
6/26/2008	Modified scenario section. See change log for list of code changes.	Georgia Tech	0.22
7/11/2008	Significant rewrite of the documentation. See change log for list of code changes.	Georgia Tech	0.3
8/7/2008	Changed the syntax for http storeEvent and storeSample requests. Added LDAP functionality. See change log for list of code changes.	Georgia Tech	0.4
8/15/2008	SHDR rejecting Event other than type Alarm. Various other fixes. See change log for list of code changes.	Georgia Tech	0.41
10/20/2008	Changed Adapter to be a MS service for the Core Agent and the SHDR Agent.	Georgia Tech	0.45
12/22/2008	Revamped the Agent SDK document for clarity and further definition.	Georgia Tech	0.5
12/8/2010	Updated for MTConnect version 1.1.0	Georgia Tech	2.1
03/11/2011	Updated by Lei Yang version 2.1.7	Georgia Tech	2.1.7

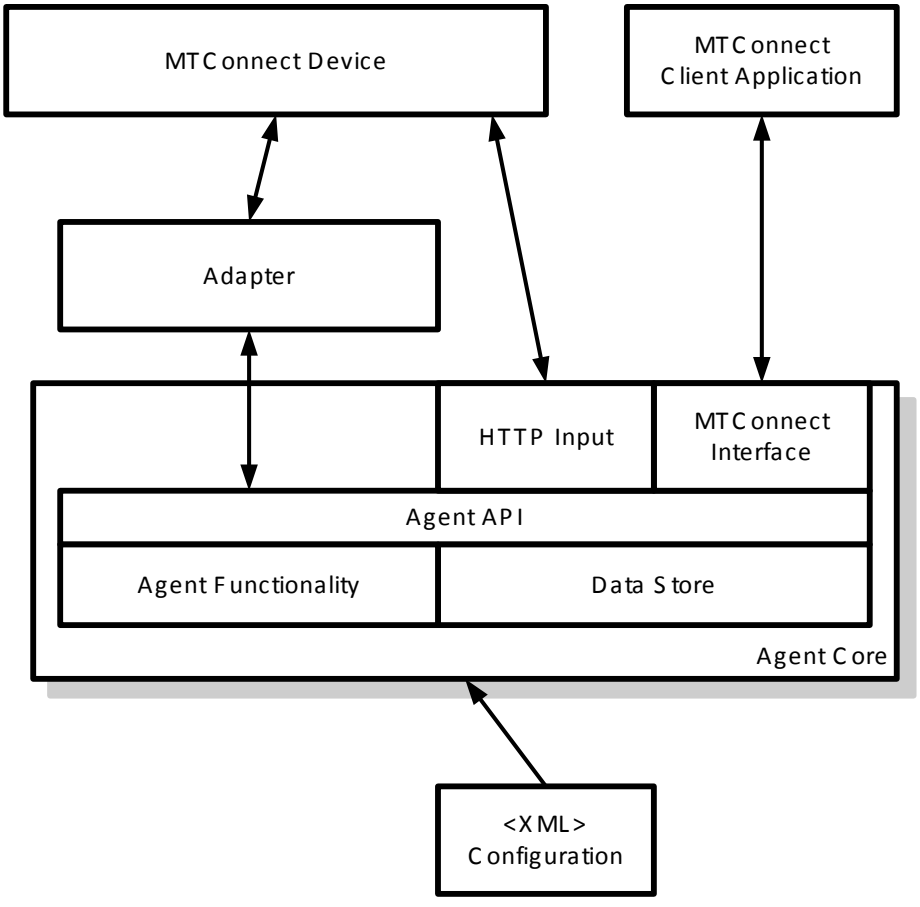
Introduction

The .NET software development kit (SDK) was written to reduce the time and effort required to implement an MTConnect interface. It consists of four Visual Studio Projects that allow implementers to use the SDK in a variety of scenarios.

The SDK was developed in Visual Studio 2008 and written in C#. It can be embedded directly into your application, used as a stand-alone Windows application or run as a Windows service.

Architecture

The following diagram provides an overview of the .NET Agent SDK architecture:



The Agent Core of the SDK contains the base functionality of the agent and can be accessed via function calls or HTTP. Functional and HTTP access allow the SDK to be used in a variety of ways

SDK Solution and Projects

The SDK is written in Microsoft Visual Studio 2008 using C# and consists of one Visual Studio Solution containing four Visual Studio Projects. The solution and projects are as follows:

MTConnectAgent Solution
This solution contains the four SDK projects.

MTConnectAgentCore Project
This project produces an Agent Core DLL that can be embedded into an application.

MTConnectAgentCoreWindows Project
This project produces an executable file that exercises the Agent Core DLL through a simple graphical user interface.

MTConnectAgentCoreWindowsService Project
This project defines the MTConnect Agent Core as a window service.

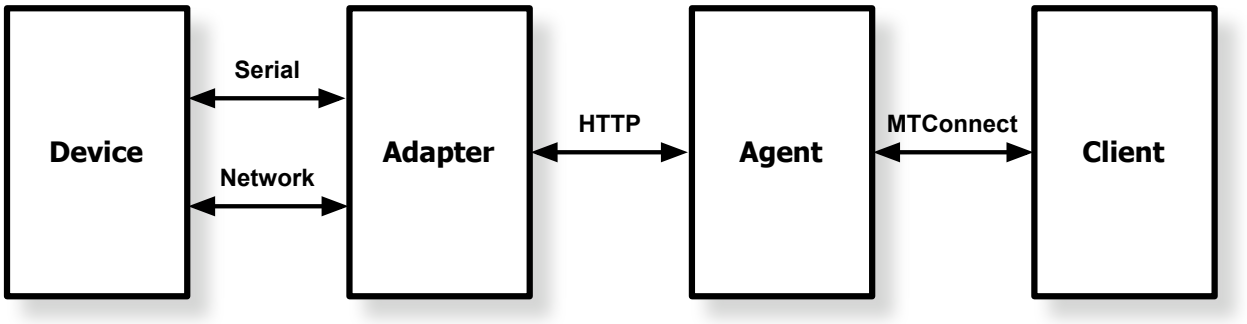
MTConnectAgentCoreWindowsServiceSetup Project
This project produces an install shield and a setup.exe for the Core Agent Service.

Implementation Scenarios

This section describes three possible scenarios in which the SDK can be used to implement an MTConnect interface.

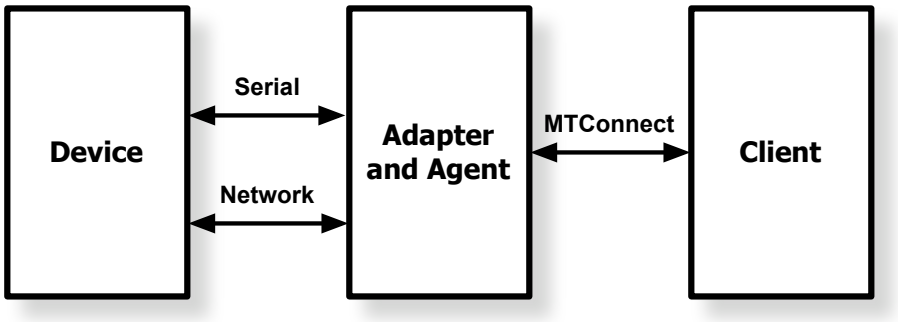
Independent Adapter and Independent Agent

In this scenario, the adapter and agent are implemented as independent entities and can be manifested as either Windows Applications or Windows Services. The adapter communicates with the controller (typically via a serial or network connection) and with the agent via http. The agent provides the MTConnect interface to the client. The following diagram illustrates this scenario:



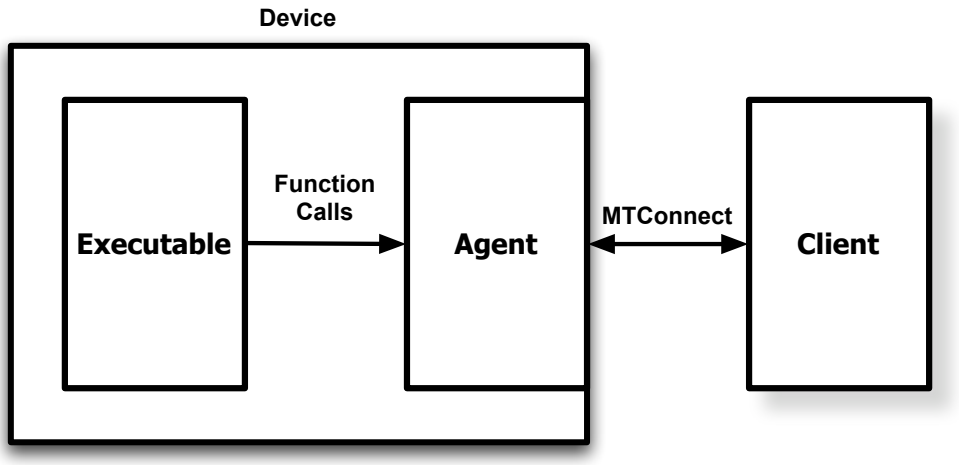
Combined Adapter and Agent

In this scenario, the adapter and agent are combined into a single Windows Application or Windows Service. The combined entity communicates with both the device and the MTConnect client.



Embedded Agent

In this scenario, the agent is embedded in the device and the device communicates with the agent through function calls. The agent communicates with the MTConnect client. The following diagram illustrates this scenario:



How to Guide

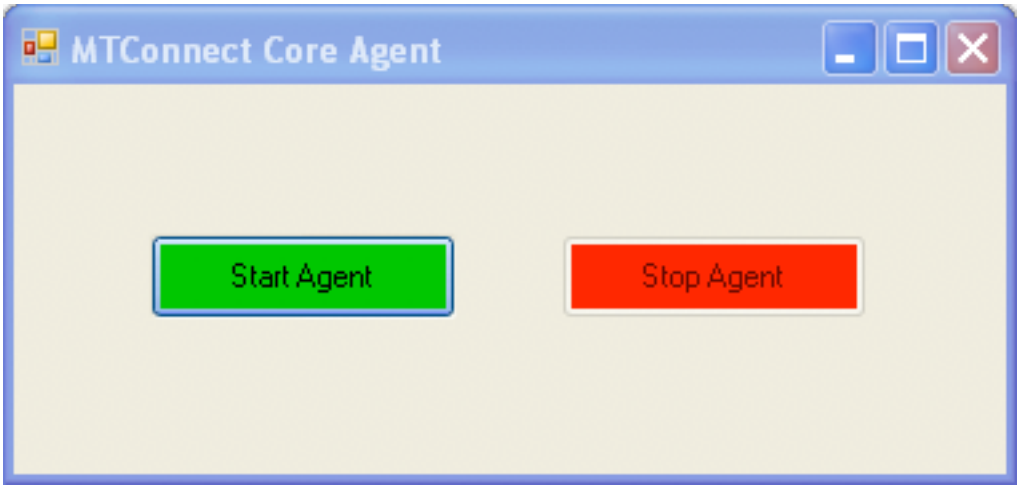
Windows Application Version

How do I install the Windows Application Version?

1. Unzip the SDK file in a known location.
2. Make sure the Microsoft .NET Framework 3.5 (or later) is installed on your computer.

How do I run the Windows Application Version?

1. Edit the devices.xml and mtcagent.xml files in the \src\MTConnectAgentCoreWindows\bin\Release directory to meet your needs. Information about the configuration file can be found in the Configuration File section of this document.
2. Run MTConnectAgentWindows.exe located \src\MTConnectAgentCoreWindows\bin\Release
3. The following form will appear:



4. Click on 'Start Agent' to start the agent.
5. You can now interact with the agent through its web interface.
6. Click on 'Stop Agent' to stop the agent.

Windows Service Version

How do I install the Windows Service Version?

1. Log on to the target computer with an account that has administrator privileges.
2. Make sure the Microsoft .NET framework version 3.5 (or later) is installed.
3. Uncompress the zip file that was included with this distribution.
4. Run setup.exe in the src\MTConnectAgentWindowsServiceSetup\Release directory to install the service version.
5. Edit the devices.xml and mtcagent.xml files in the install directory to meet your needs. Information about the configuration file can be found in the Configuration File section of this document.
6. You may have to reboot the computer.

How to Run

The MTConnect Agent Windows Service runs as a Microsoft windows service. Hence, it will start automatically when the computer starts. To proactively start the MTConnect Agent, please do the following:

1. Click Start on the Windows task bar.
2. Click Control Panel
3. Double click Administrative Tools
4. Double click Services
5. Click on the MTConnect entry
6. Click Start

How to Stop

To stop the MTConnect Agent, please do the following:

1. Click Start on the Windows task bar.
2. Click Control Panel
3. Double click Administrative Tools
4. Double click Services
5. Click on the Agent entry
6. Click Stop

How to Uninstall

To uninstall the MTConnect Agent, please do the following:

1. Click Start on the Windows task bar.
2. Click Control Panel
3. Double click Add or Remove Programs
4. Click Agent
5. Click Remove

Agent Interaction

How do I send an HTTP Request to the Core Agent?

- Enter a valid MTConnect request in your browser.
- Examples include:
  - http://127.0.0.1/probe
  - http://127.0.0.1/current
- For a complete overview of MTConnect requests, please see the MTConnect standard.
- A program can also send HTTP requests.

How do I send Data to the Core via HTTP?

- Enter the following in your browser:
  http://<agentAddress>/storeSample?<parameter 1>=<value 1>
  &<parameter 2>=<value 2>&<parameter n>=<value n>
- An example:
  http://127.0.0.1/storeSample?timestamp=2008-09-04T11:34:32-04:00
  &deviceName=Device1&dataItemId=Srpm&value=0.8256
- Please see the HTTP storeSample section for specifics.

How do I configure the Core Agent?

When the Agent starts, it reads the devices.xml file and configures itself based upon the contents of the file. To configure the agent:

- Stop the Agent through the GUI.
- Edit the Devices.xml file that is located in the same directory as the executable file. Please note that the Devices.xml file must adhere to the MTConnectDevices.xsd schema.
- Start the Agent.

How can I view data stored by the Agent?

- Use the http debug request.
- An example: http://127.0.0.1/debug.

Development

How can I Use the DLL in a Visual Studio 2005 C# Project?

- Start your C# project in Visual Studio 2005
- Click Project menu.
- Click Add Reference.
- Click Browse tab.
- Navigate to the location of the MTConnectCoreAgent.DLL
- Highlight the DLL
- Click OK.
- Add a reference to the MTConnect Agent Core namespace in your project:
  ```
  using MTConnectAgentCore;
  ```
- Instantiate an instance of the Agent object:
  ```
  Agent agent;
  ```
- Invoke the object constructor:
  ```
  agent = new Agent();
  ```
- Invoke the object methods.
  Example:
  ```
  agent.Start();
  ```

Function Interface

The Core Agent functionality can be embedded into an application by including the Core Agent Microsoft Dynamic Link Library (DLL), MTConnectAgentCore.dll. The DLL is located in the bin\Release directory of the MTConnectAgentCore project. The public methods available through the DLL are as follows:

Start Function

Name	Start
Description	Used to start the agent.
Declaration	`public void Start()`
Throws Exception	**AgentException**(**string** *msg*, **System.Exception** *innerException*)

Stop Function

Name	Stop
Description	Used to stop the agent.
Declaration	`public void Stop()`
Throws Exception	**AgentException**(**string** *msg*, **System.Exception** *innerException*)

StoreSample Function

Description	Used to store data in the agent repository.	
Declaration	`public short StoreSample(String timestamp, String dataItemId, String value)`	
Return Values	0	The function completed successfully.
	1	The function completed successfully, but a warning message was generated.
	2	The function did not complete successfully, and an error message was generated.

Attributes		
Name	Description	Usage
`timestamp`	The message time stamp.	A valid w3c date and time format or an empty string is required. If an empty string is passed, the agent will generate the value.
`dataItemId`	The dataItemId value.	A valid string is required that matches a dataItemId contained in the devices.xml file.
`value`	The data value.	Any valid string can be used. NULL cannot be used.

StoreEvent Function

Description	Used to store event data in the agent repository.	
Declaration	`public short StoreEvent(String timestamp, String dataItemId, String value, String code, String nativeCode)`	
Return Value Meanings	0	The function completed successfully.
	1	The function completed successfully, but a warning message was generated.
	2	The function did not complete successfully, and an error message was generated.

Attributes		
Name	Description	Usage
`timestamp`	The message time stamp.	A valid w3c date and time format or an empty string is required. If an empty string is passed, the agent will generate the value.
`dataItemId`	The dataItemId value.	A valid string is required that matches a dataItemId contained in the devices.xml file.
`value`	The event value.	Any valid string can be used. NULL cannot be used.
`code`	The event code.	Any valid string can be used.
`nativeCode`	The native code for the piece of equipment. This is the way the event is represented on the component.	Any valid string can be used.

StoreCondition Function

Description	Used to store condition data in the agent repository.	
Declaration	`public short StoreCondition(String timestamp, String dataItemId, String condition, String value, String nativeCode)`	
Return Value Meanings	0	The function completed successfully.
	1	The function completed successfully, but a warning message was generated.
	2	The function did not complete successfully, and an error message was generated.

Attributes		
Name	Description	Usage
`timestamp`	The message time stamp.	A valid w3c date and time format or an empty string is required. If an empty string is passed, the agent will generate the value.
`dataItemId`	The dataItemId value.	A valid string is required that matches a dataItemId contained in the devices.xml file.
`condition`	The condition name.	Valid strings include: Unavailable, Normal, Warning, or Fault
`value`	The condition value.	Any valid string can be used.
`nativeCode`	The native code for the piece of equipment.	Any valid string can be used.

HTTP Interface

Interaction with the agent can also be accomplished exclusively through HTTP.

The agent supports the standard HTTP requests:

- Probe, MTConnect probe request
- Current, MTConnect current request
- Sample, MTConnect sample request

In addition, it supports the following HTTP requests:

- Debug, Used to obtain a listing of data stored in the Agent's repository.
- Store Sample, Used to provide data samples to the Agent.
- Store Event, Used to provide Event data to the Agent.
- Store Condition, Used to provide Condition data to the Agent.
- Version, Lists the agent version.
- /, Indicates if the agent is running.

For specifics about the MTConnect probe, current and sample requests please see the MTConnect specification.

Requests to the agent are provided as HTTP strings. Specifics about the strings are as follows:

- The parameters can be provided in any order.
- Unrecognized parameters will be ignored.
- Line breaks are not allowed.
- Encode spaces with "+".
- Absence of a value means the parameter value is empty string.
- Absence of a parameter means the parameter value is NULL.
- Do not enclose values in quotes.
- References:
    - http://en.wikipedia.org/wiki/Query_string
    - http://tools.ietf.org/html/rfc1738

Example:

http://127.0.0.1/storeSample?timestamp=2008-09-22T11:14:42:00-04:00&dataItemId=myItem&value=0.8256

debug

The HTTP debug request displays the data stored in the agent. The syntax of the debug request is as follows:

http://<agent Address>/debug

Where, <agentAddress> = IP# or DNS of Agent (required)

storeSample

The storeSample HTTP request allows sample data to be sent to the Agent. The parameters for the request are as follows:

Name	Description	Usage
timestamp	The message time stamp.	A valid w3c date and time format or an empty string is required. If an empty string is passed, the agent will generate the value.
dataItemId	The dataItemId value.	A valid string is required that matches a dataItemId contained in the devices.xml file
value	The data value.	Any valid string can be used. NULL cannot be used.

Example:

http://127.0.0.1/storeSample?timestamp=2008-04-30T15:26:17-04:00&dataItemId=VROOM1&value=0.8256

storeEvent

The storeEvent HTTP request allows event data to be sent to the Agent. The parameters for the request are as follows:

Name	Description	Usage
timestamp	The message time stamp.	A valid w3c date and time format or an empty string is required. If an empty string is passed, the agent will generate the value.
dataItemId	The name of the dataItemId.	A valid string is required that matches a dataItemId contained in the devices.xml file.
value	The event value.	Any valid string can be used. NULL cannot be used.
code	The event code.	Any valid string can be used. NULL cannot be used.
nativeCode	The native code for the piece of equipment. This is the way the event is represented on the component.	Any valid string can be used. NULL cannot be used.

Example:

http://127.0.0.1/storeEvent?timestamp=2008-04-30T15:26:17-04:00&dataItemId=event12&value=good&code=OTHER&nativeCode=samplenativeCode

storeCondition

The storeCondition HTTP request allows condition data to be sent to the Agent. The parameters for the request are as follows:

Name	Description	Usage
timestamp	The message time stamp.	A valid w3c date and time format or an empty string is required. If an empty string is passed, the agent will generate the value.
dataItemId	The name of the dataItemId.	A valid string is required that matches a dataItemId contained in the devices.xml file.
condition	The condition of the device.	Valid strings include: "Normal", "Warning", "Fault" or "Unavailable".
value	The condition value.	Any valid string can be used. NULL cannot be used.
code	The condition code.	Any valid string can be used.
nativeCode	The native code for the piece of equipment. This is the way the event is represented on the component.	Any valid string can be used.

Example:

http://127.0.0.1/storeCondition?timestamp=2008-04-30T15:26:17-04:00&dataItemId=condition1&value=Unavailable&condition=Unavailable&code=samplecode&nativeCode=samplenativeCode

version

The HTTP version request displays the version of the agent. The syntax of the request is as follows:

http://<agent Address>/version

Agent Configuration

Upon startup, the agent reads configuration values from mtcagent.ini. Mtcagent.ini is an xml file, adheres to the mtcagent.xsd schema and contains the elements listed in the table below. Mtcagent.ini and mtcagent.xsd are located in the same directory as the exe or DLL file being used.

Configuration File Elements		
Element Name	Description	Usage
CreateNewLogFile	Should the old log file be deleted and new on started?	true or false (case insensitive)
UseLogFile	Should debug items be logged to a file?	true or false (case insensitive)

SDK Package Structure

The MTConnect Client SDK is delivered as a zip file that contains the following directories:

General

\
This is the root directory that contains a readme file. The readme file contains a brief description of the directory listing.

\doc
This directory contains the SDK documentation.

\src
This is the source directory that contains the Visual Studio solution and projects.

Core Agent

\src
This is the solution directory. It contains the MTConnectAgent.sln file and the project directories.

\src\MTConnectAgentCore
This directory contains the MTConnectAgentCore project
\src\MTConnectAgentCore\bin\Release
This directory contains the MTConnectAgentCore.DLL which is produced by the MTConnectAgentCore project.

Windows Executable to Run Core Agent

\src\MTConnectAgentCoreWindows
This directory contains the MTConnectAgentWindows project.

\src\MTConnectAgentCoreWindows\bin\Release
This directory contains the MTConnectAgentWindows.exe file which is produced by the MTConnectAgentWindows project.

Windows Service to Run Core Agent

\src\MTConnectAgentWindowsService
This directory contains the MTConnectAgentWindowsService project which defines the MTConnect Agent Core as a window service.

\src\MTConnectAgentWindowsService\bin\Release
This directory contains the MTConnectAgentWindowsService.exe file which is produced by the MTConnectAgentWindowsService project.

Create Core Agent Windows Service

\src\MTConnectAgentWindowsServiceSetup
This directory contains the MTConnectAgentWindowsServiceSetup project which creates the MTConnect Agent Core window service install.

\src\MTConnectAgentWindowsServiceSetup\Release
This directory contains the MTConnectAgentWindowsService.msi file and setup.exe.

Known Issues

The SDK does not support the frequency parameter for the sample request.

Comments or Questions

The Factory Information Systems Lab at Georgia Tech (www.fis.gatech.edu) developed the agent. For further information, please contact:

Andrew D. Dugenske
Manufacturing Research Center
Georgia Institute of Technology
Atlanta, GA 30332-0560
USA
dugenske@gatech.edu
+1 404 894 9161