README

Friday, August 05, 2016

This document presents a brief background on the mechanics of the MTConnect agent for the ABB robot. This document assumes the reader is familiar with MTConnect operation, and for deeper explanation of MTConnect, please refer to MTConnect URL: <http://www.mtconnect.org/> for more information. This document concerns itself with how an embedded Adapter operates. One or more embedded adapters read a configuration file, and then read a remote ABB robot log files as logged from an ABB robot device. Boeing has writtern a log file mechanism for the ABB robot machines, that must be enabled, and then a log file will be created on an ftp site. This ABB robot log file must be shared in Windows so the MTConnect embedded adapter can read the file, reinterpret the contents into MTConnect meaningful terminology, and then report this information to a canned MTConnect Agent (circa version 1.2), which handles management of devices(s) data, the web service, etc. The ABB robot log file is a Comma Separated File (CSV) with the comma character as the delimiter between log fields on a single line.

# Background

MTConnect is a new standard developed to facilitate the exchange of data on the manufacturing floor. The MTConnect open specification provides for cost effective data acquisition on the manufacturing floor for machine tools and related devices. MTConnect is based upon prevalent Web technology including XML and HTTP. Figure 1 shows the MT Connect architecture. An “MTConnect Device” is a piece of equipment – in this case a ABB robot machine tool, which (optionally) includes an MTConnect Adapter so that we can get data from it. The “Agent” is a process that acts as a “bridge” between a device and a factory “Client Application”. To learn more about MTConnect visit: <http://www.mtconnect.org/>

Figure 1 shows a typical MTConnect ABB robot Logging Agent system architecture (with one or more ABB robot devices). Communication between two Windows PCs is assumed - one containing the ABB robot log file and the other PC communicating over Ethernet to read the ABB robot log file.



This document describes the data gathering for a ABB robot that updates a comma separated log file. Multiple adapters to ABB software are possible, each contained within one MTConnect Agent.

The ABB Agent contains back end adapters that read a log file generated from the periodically (typically when an event occurs within the ). The log file contains events and not samples, but all the events are time stamped and in absolute order of occurrence.

The file is specified as a Window cross-platform file, so it must contain the PC or computer name. UNC is short for Universal Naming Convention and specifies a Windows syntax to describe the location of a network resource, such as a shared file, directory, or printer. The UNC syntax for Windows systems has the generic form:

\\ComputerName\SharedResource

In our case the SharedResource is a shared file that must be explicitly sharable. In order to use the UNC file, Microsoft File Operations: CreateFile, ReadFile and CloseFile are used as other generic C++ file operations did not work (but were originally tried.) UNC files on Windows seem to require Windows specific File operations. Note, the UNC file path must be accessible to other computers or it cannot be read.mInside the Agent are Adapters for each UNC file. Each Adapter runs as a thread, hence the distinction between 64 bit and 32 bit C++ solutions must be explicitly acknowledged in installing the binary exe. That is, 32-bit MTConnect agents do not on 64 bit platforms, although they may appear to.

The ABB robot log file encoding is:

/\*\*

2016-04-05 17:11:47,TRUE,100,FALSE,testName1,847.225,-1052.83,685.211\_

0 - date

1- ON

2- override

3 - ?

4 - program name

5 - x

6 - y

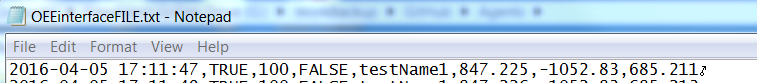
7 – z

8 – character ‘0x0B'

\*/

|  |  |
| --- | --- |
| Log file State | MTConnect Interpretation |
| Power = FALSE | All off |
| Power= True | Avail=AVAILABLE  CONTROLLERMODE=AUTOMATIC (assume\_)  Power=ON |
| Program=text | PROGRAM =""  EXECUTION= EXECUTING  CONTROLLERMODE=AUTOMATIC |
| Program=blank | PROGRAM =""  EXECUTION=IDLE  CONTROLLERMODE=MANUAL |
| EXECUTION= EXECUTING  CONTROLLERMODE=AUTOMATIC | Xabs,Yabs,Zabs incremented  Srpm =99  Has to be moving to be “busy” |
| EXECUTION=IDLE  CONTROLLERMODE=MANUAL | Xabs,Yabs,Zabs sme  Srpm =0 |
| XYZ | Updated to Xabs,Yabs,Zabs when read |
| Override | Set to path\_feedrateovr |
|  |  |

There is a trailing arrow character – not sure if intentional.



I force the xyz to move and have an rpm > 0 if the controller is in automatic and executing.

## Config ftp log file sharing

In the config.ini file there are fields under each device specification for User, Pw. So for Device M1, you will need to fill in the fields if you want an ftp access. Note the user will need the domain appended to the front (i.e., “NIST\”). The ProductionLog must start with the ftp: qualifier or it won't be identified.

[M1]

ProductionLog= ftp://localhost/OEEinterfaceFILE.txt

QueryServer=20000

ServerRate=2000

User=NIST\michalos

Pw=REALPASSWORDGOESHERE

[M2]

…

Can have different specification under the [M2] device section

Below corresponds to the beginning of the config.ini file containing:

[GLOBALS]

Config=UPDATED

ServiceName=ABBAgent

AgentPort=5020

QueryServer=10000

ServerRate=5000

Debug=0

MTConnectDevice=M1,M2

logging\_level=FATAL

# Describe data format 08/10/2015 – only good for

DateFormat=dd/mm/yyyy

# Now come device ini sections such as [M1]

If these config.ini fields (i.e., User, Pw, LocalShare and NetworkShare) are set, then the code will attempt to log into the ftp site with the given credentials.

WIN32\_FIND\_DATA FileData;

HINTERNET m\_hInternet;

HINTERNET m\_hConnection;

bool overwrite=true;

std::string ip = unc.substr(6); // skip ftp://

ip = ip.substr( 0, ip.find\_last\_of( '/' ) );

std::string remoteFile = unc.substr( unc.find\_last\_of( '/' ) + 1 );

std::string localFile = File.TempFileName( \_device.c\_str(), 99);

m\_hInternet = InternetOpen(("MTConnect" + \_device).c\_str(), INTERNET\_OPEN\_TYPE\_DIRECT, 0, 0, 0);

m\_hConnection = InternetConnect(m\_hInternet,ip.c\_str(), INTERNET\_DEFAULT\_FTP\_PORT, \_User.c\_str(), \_Password.c\_str(), INTERNET\_SERVICE\_FTP, 0, 0);

// Use for computing last modification time

FtpFindFirstFile(m\_hConnection, remoteFile.c\_str(), &FileData, INTERNET\_FLAG\_NEED\_FILE, NULL);

COleDateTime filemodtime(FileData.ftLastWriteTime);

if( lastftpmodtime == filemodtime)

{

InternetCloseHandle(m\_hConnection);

InternetCloseHandle(m\_hInternet);

return WarnWithMsg(E\_PENDING, StdStringFormat("ftp remote log file Device %s modtime unchanged %s\n", \_device.c\_str( ), lastmodtime.Format("%c") ) );;

}

FtpGetFile(m\_hConnection, remoteFile.c\_str(), localFile.c\_str(), (BOOL)overwrite, FILE\_ATTRIBUTE\_NORMAL, FTP\_TRANSFER\_TYPE\_UNKNOWN, 0);

lastftpmodtime=filemodtime;

InternetCloseHandle(m\_hConnection);

InternetCloseHandle(m\_hInternet);

filename=localFile;

## Configure to add new Devices

To add a new device change [GLOBALS].Config to NEW and add the device(s) to [GLOBALS]. MTConnectDevice

[GLOBALS]

Config=NEW

sleep=2000

ResetAtMidnite=false

Debug=0

MTConnectDevice=M1,M2

ServiceName=ABBAgent

logging\_level=FATAL

QueryServer=10000

ServerRate=5000

[M1]

ProductionLog=ftp://ftphost/OEEinterfaceFILE1.txt

[M2]

ProductionLog=ftp://ftphost/OEEinterfaceFILE2.txt

If the update works, [GLOBALS].Config = UPDATED will be reported in the config.ini file. If not, [GLOBALS].Config=ERROR will be reported in the config.ini file.

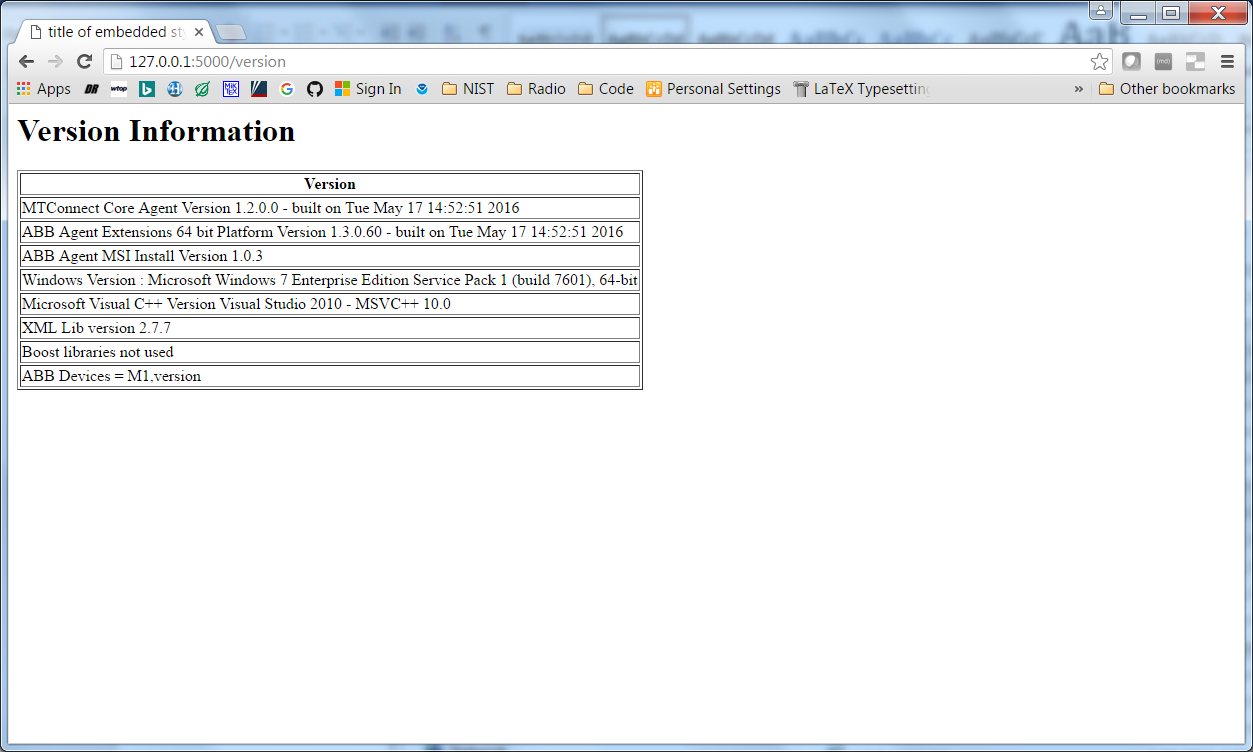
# Version Information from MTConnect Agent

The version information of the various MTConnect components is available through web browser access. It is recommended to use Google Chrome as it understand the XSLT formatting of the XML (Internet Explorer is baffled.)

The open source core MTConnect 1.2 Agent is downloaded from github and “frozen”. The compiled version used is:

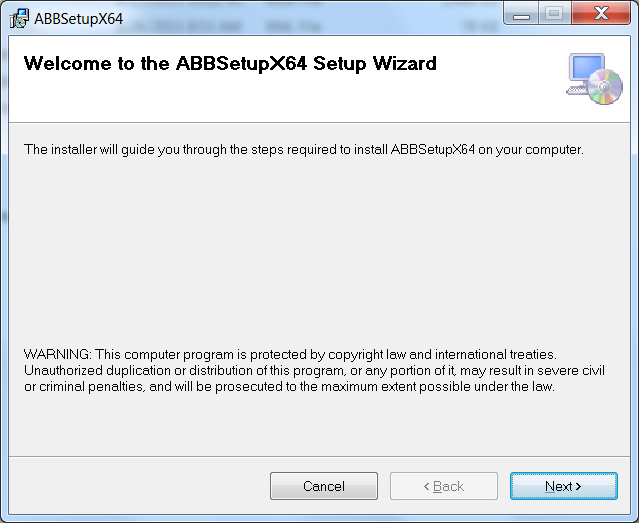
MTConnect Core Agent Version 1.2.0.0 - built on Tue May 17 14:52:51 2016

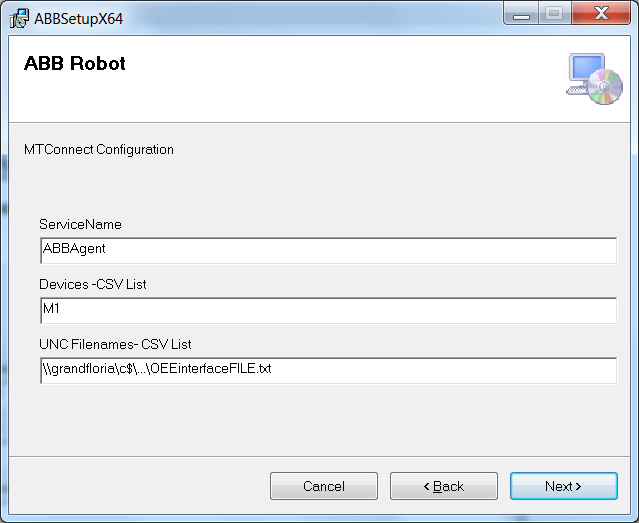
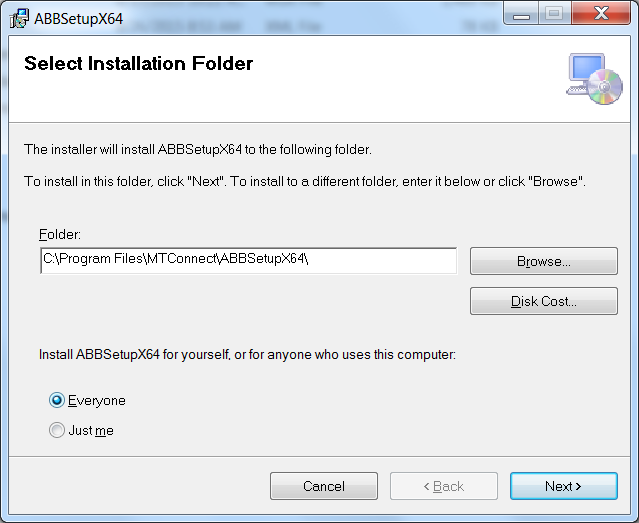
It is not perfect but the output shown below gives an indication of the software involved in the MTConnect Agent operation. It can be modified to include other version information, but requires a recompilation at this time.

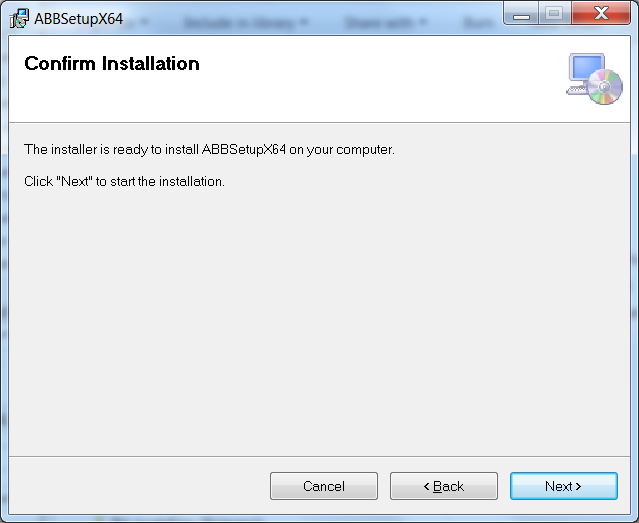


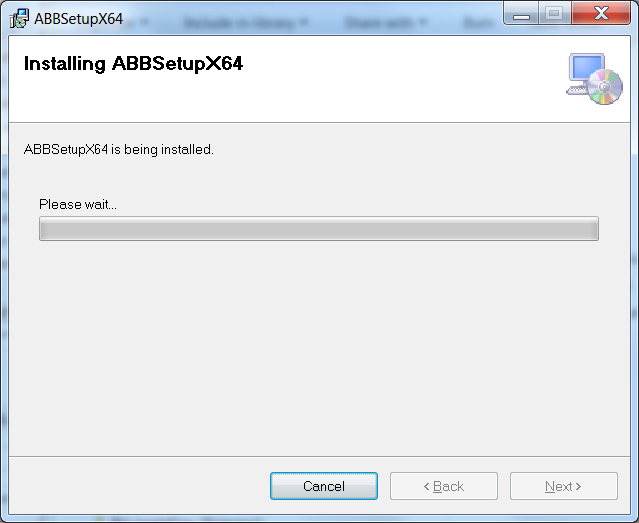
# Installation

To install the ABB Agent double click the (for 64 bit machines only).

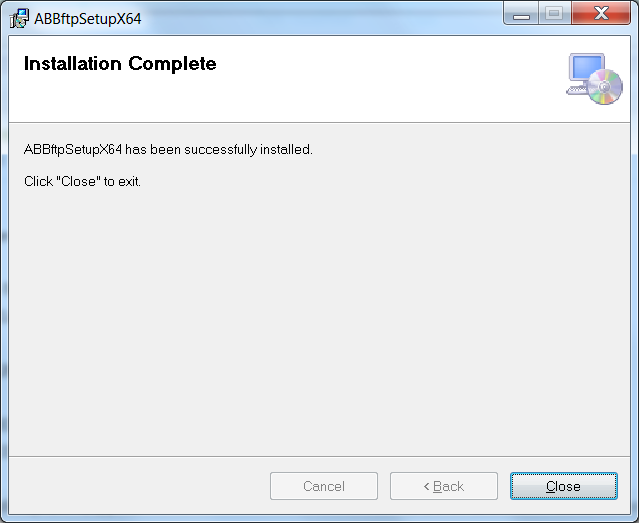




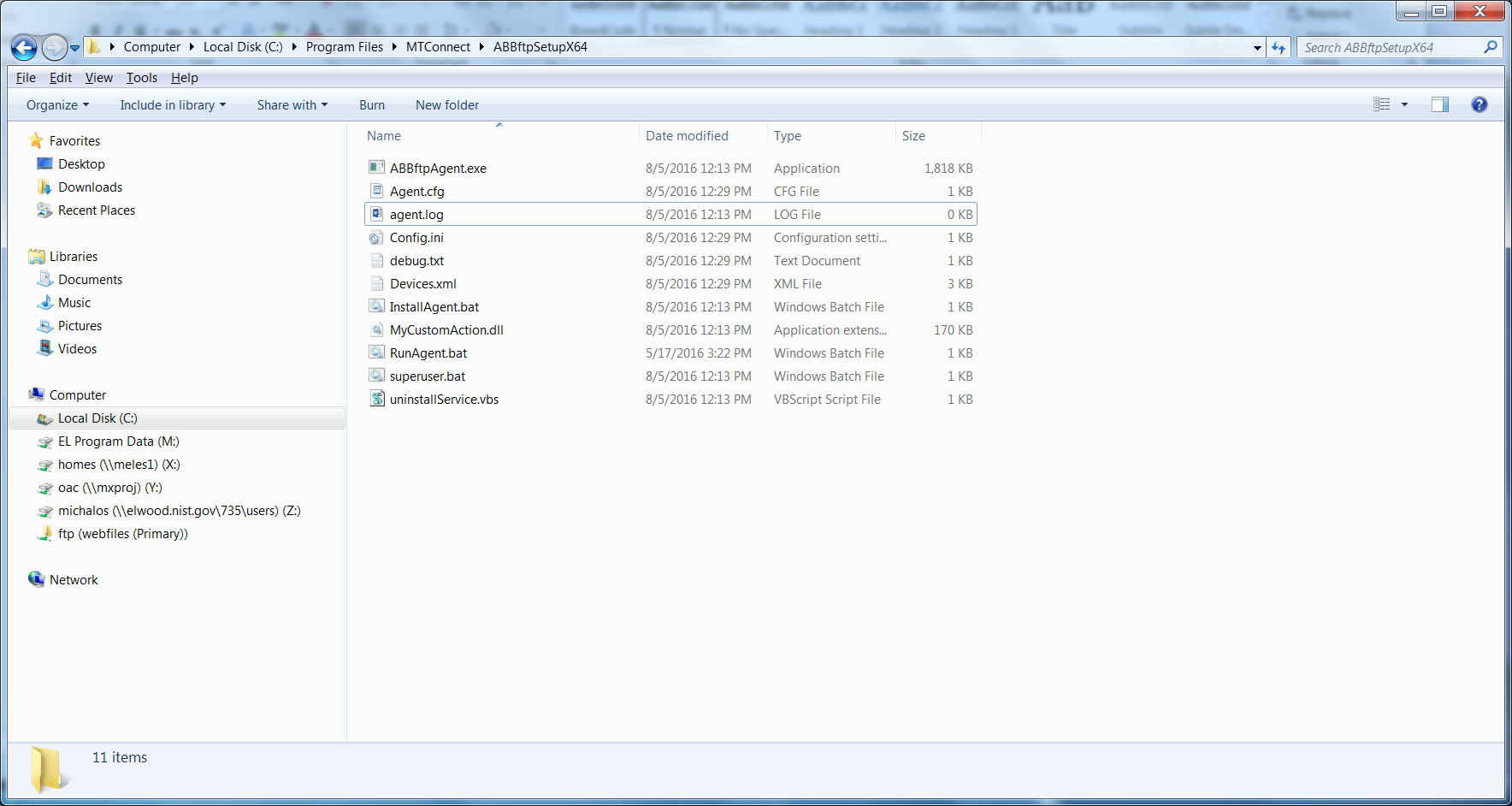




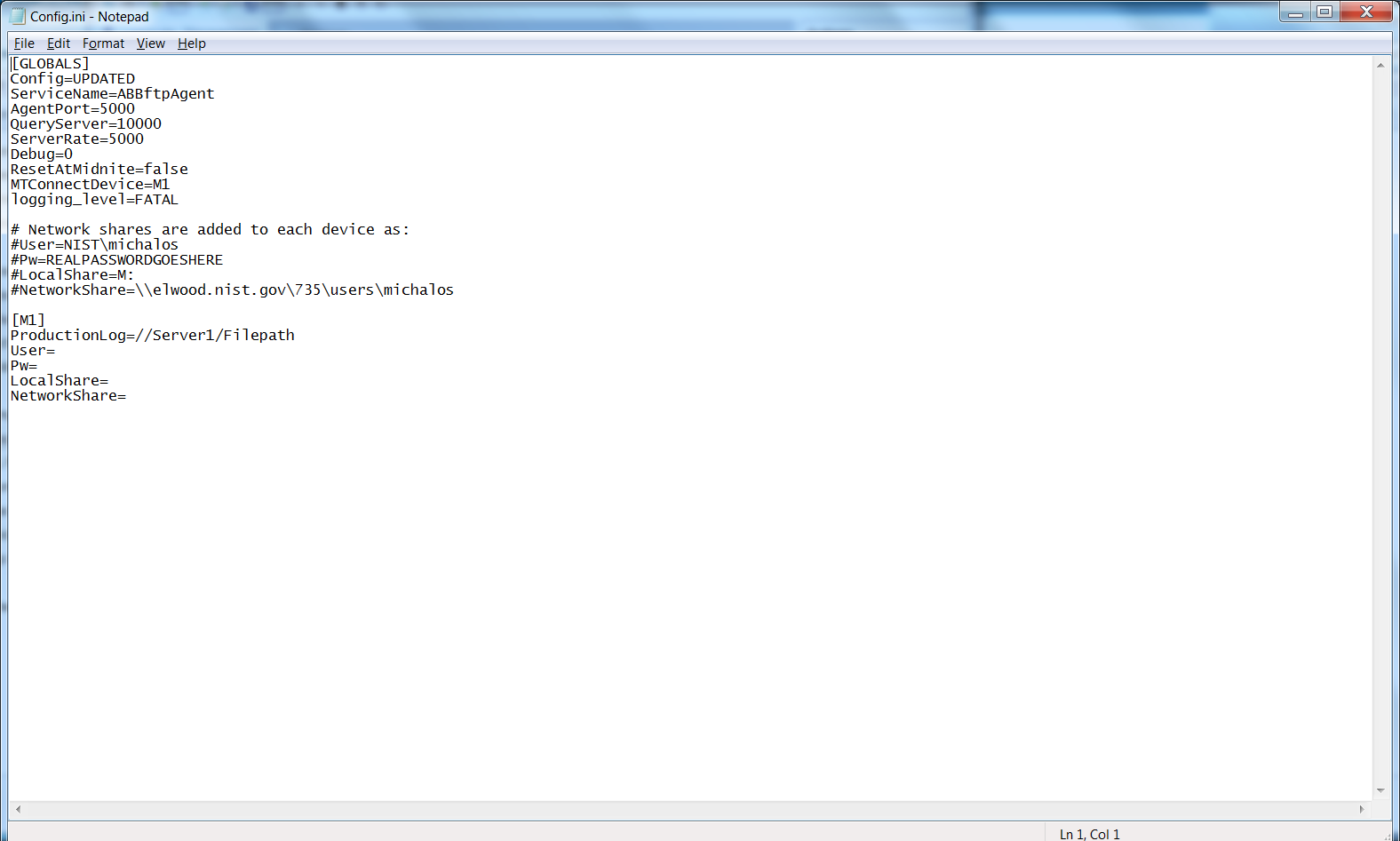
Acknowledge the installation permission challenge, then you should see the installation screen:

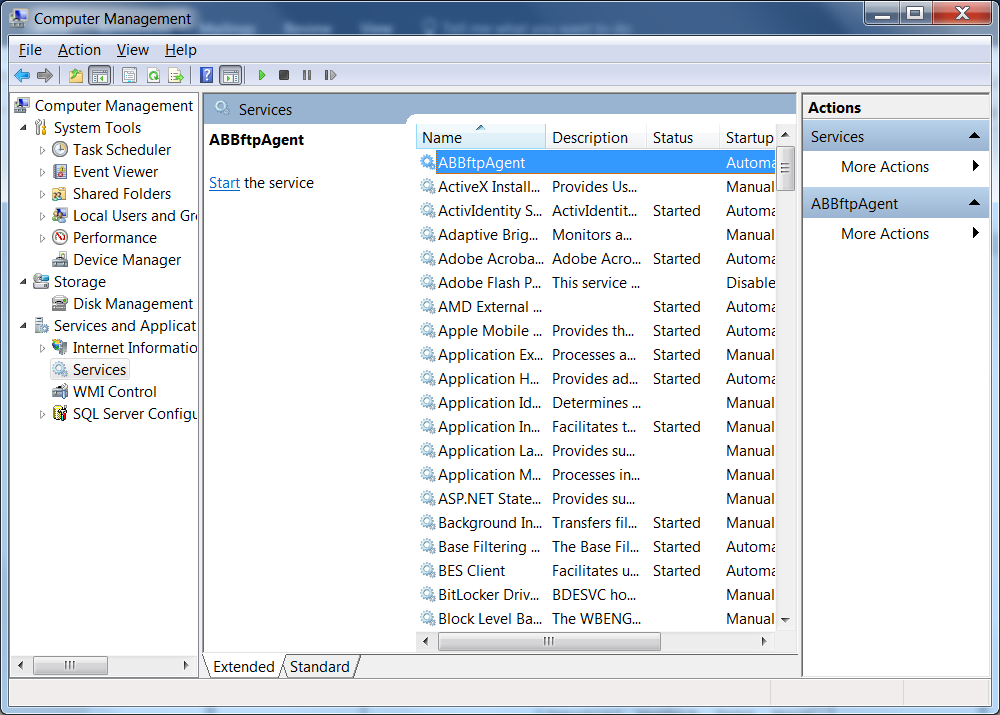


You can modify the setup by navigating to:



And then opening the file config.ini: (read the section on configuration for modifying).





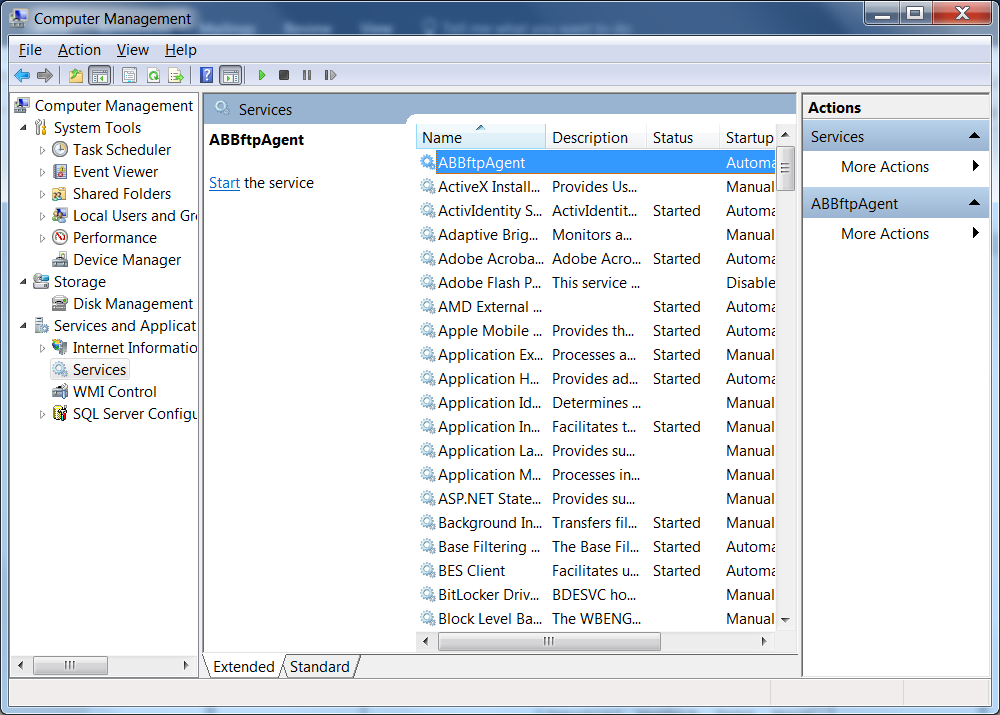
The agent will NOT start unless you tell it to START (unless you reboot).

# Uninstall

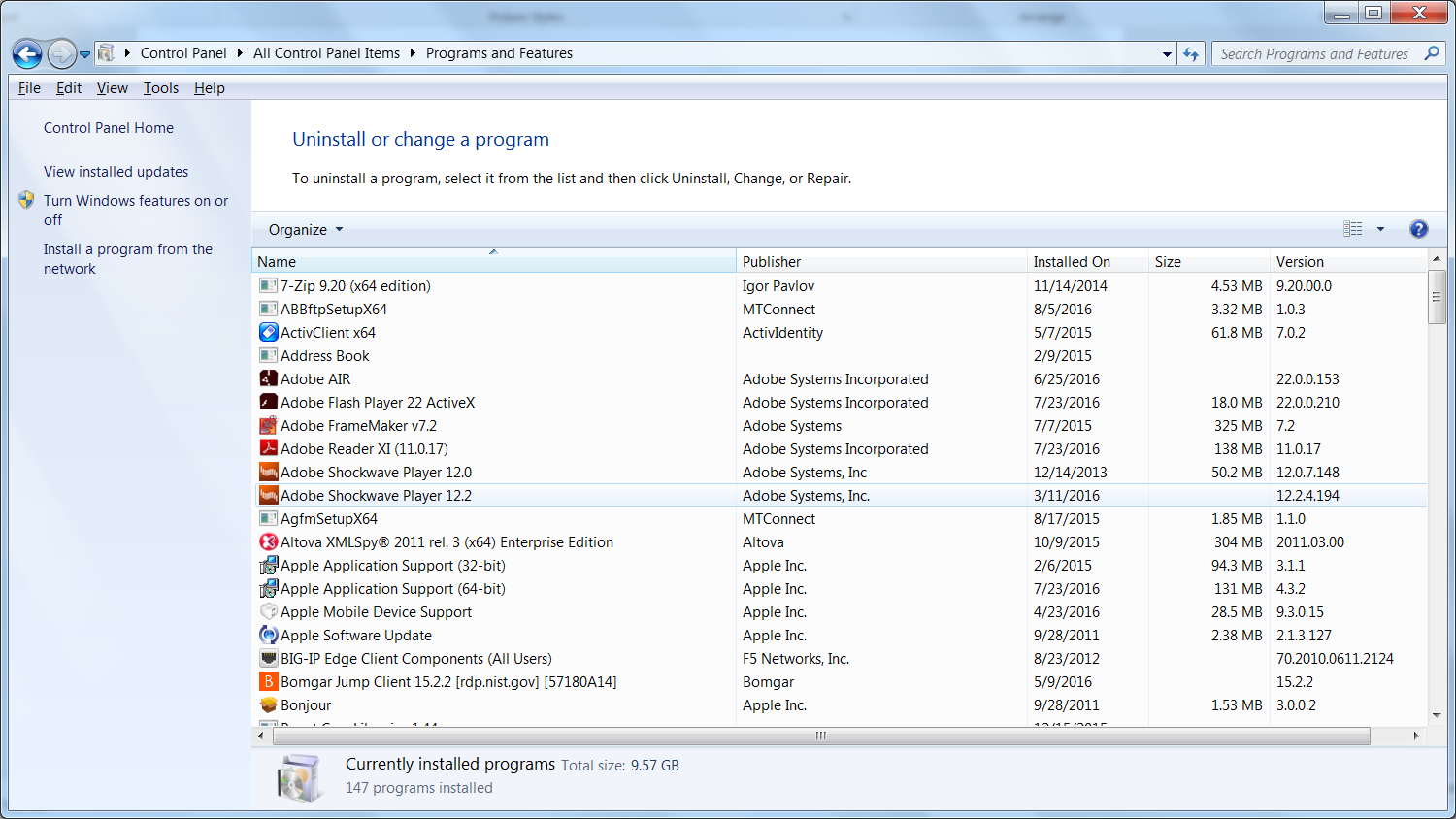
First make sure the ABB Agent is stopped in the Service Control Manager:

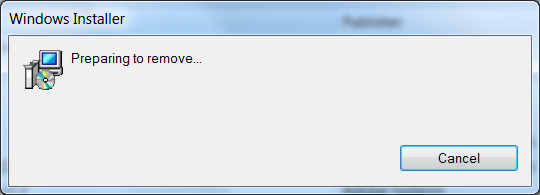
Right click on My Computer -> Select Manage -> Acknowledge UAC challenge

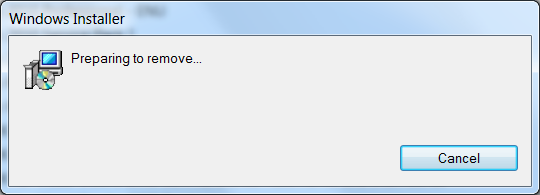
Select Services and Applications and then Services, scroll down to ABBAgent, and push Stop button.



After you have stopped the ABB Agent service, go into the control panel and uninstall the program: MTConnectAgentABBx64





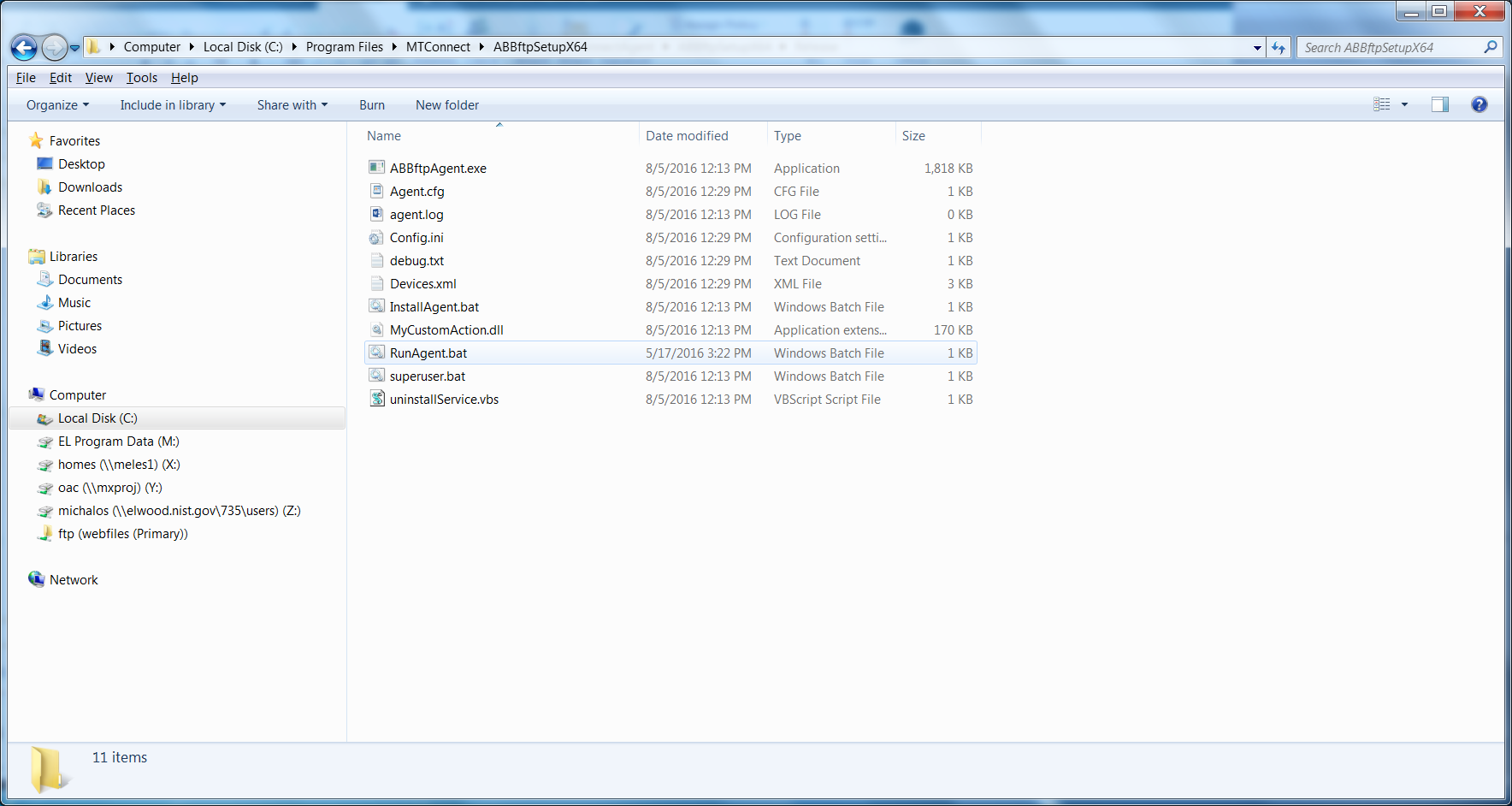


Please wait .. acknowledge UAC permission to uninstall challenge (you must be administrator or have administrator priviledges). It will uninstall and you may see a black console screen popup in the background momemtarily.

And then in the Service Manager click Action->Refresh, and the ABB Agent service should be removed. The agent code in C:\Program Files\MTConnect\MTConnectAgentABBx64 should also be removed.

# Configuration

The installation wizard installs the log file Agent into the folder: C:\Program Files\MTConnect\MTConnectAgentABBx64 where x64 means a 64 bit installation platform, e.g., windows 7. The MTConnect Agent executable must be paired with the correct platform (32 or 64 bit).



In the folder, the vb script MTConnectPage-1.4.vbs in the can be used to verify that the log file Agent is working. MTConnectPage-1.4.vbs reads the data from the agent via <http://127.0.0.1:5000> and then formats the data. (assuming you have configure the agent port to 5000).

# Configuration

The ABB MTConnect Agent uses the open-source MTconnect agent version 1.2 as the backbone to read http requests, generate XML responses, and update the underlying data. There is a NIST log file adapter addition to remotely read local or remote windows UNC files and then parse. The adapter can handle multiple device log file and update the core MTConnect agent.

The configuration is done at installation time during the wizard screens. The user must supply pairs of device names and UNC log file name, e.g., camio1,\\servername\C$\Program Files\ABB\logfile.log From Wikipedia: The Microsoft Windows UNC, short for Universal Naming Convention or Uniform Naming Convention, specifies a common syntax to describe the location of a network resource, such as a shared file, directory, or printer. The UNC syntax for Windows systems has the generic form:

\\ComputerName\SharedFolder\Resource

Microsoft often refers to this as a "network path". You can use local paths, but if you are doing remote access of the log file, you must use the UNC convention.

The user can modify the config.ini file in the C:/Program Files/MTConnect/MTConnectAgentABBx64 folder. However, the devices.xml and the agent.cfg files necessary for the core MTConnect agent are generated at installation time, and thus a reinstallation would be required to modify these files.

In the config.ini file, you can change the ServiceName, Agent port and query times of the log files. These changes will take if you stop/restart the Agent service or reboot the machine.

[GLOBALS]

ServiceName=ABBAgent

AgentPort=5000

QueryServer=10000

ServerRate=5000

Debug=0

ResetAtMidnite=false

MTConnectDevice=M1

logging\_level=FATAL

[M1]

ProductionLog=\\servername\c

Thus, the agent it reads the config.ini file for list of “MTConnectDevice”s under the Globals section. The configuration file also sets the Global flags: QueryServer, ServerRate, Debug, AgentPort(default 5000), and ResetAtMidnite.

# DIRECTIONS TO CONFIGURE ABB LOG FILE AGENT.

Modify Config.ini in C:\Program Files\MTConnect\MTConnectAgentABBx64directory

1. Stop ABB agent, edit config.ini file, add new configuration:

[GLOBALS]

Config=NEW

1. Add new devices under [GLOBALS] section tag “MTConnectDevice” (spaces are stripped out)

MTConnectDevice=M1, M2, M3

1. Make sure there is an ini file “section” for each device (in this case M1, M2, M3) and ProductionLog tag that points to the UNC (Windows Universal Naming Convention) path to the log file as in:

[M1]

ProductionLog=\\grandflorio\c$\logfolder\Events\_log\_BP\_ABB.txt

[M2]

ProductionLog=\\rufous\c$\logfolder\Events\_log\_BP\_ABB.txt

[M3]

ProductionLog=\\synchro\c$\logfolder\Events\_log\_BP\_ABB.txt

1. Start ABB agent, the agent will detect a new configuration, and then write a new Devices.xml file to add the new devices.
2. If it works config.ini tag should say :”Config=UPDATED” if a problem tag will say: “Config=ERROR”

