README

This document describes the data gathering for an Nikon CMM machine that updates a tab separated log file. Multiple adapters to Nikon software are possible, each contained within the Agent.

The NIKON Agent contains back end adapters that read an log file generated from the CMM periodically (typically when an event occurs within the CMM). 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.

In the Nikon file, it saves all events with each line within the UNC shared file. The delimiter between fields is “tab” or the character '\08'. Each line contains an event describing a machine state transition, so that all the lines of the UNC file must be read to understand the current state (machine on is one event, and then run program is another event). Note, no error detection of runaway date or times is done.

Below is a sample of the last line found in the Nikon shared file.

```
2013-04-30 14:13:21.078    64     1024   CAMIO72u      [0001] CAMIO Studio started
```

There are five fields: timestamp, ??, ??, machine, and a status event.

Field	Example
timestamp	01/23/2014 9:48:56
??	64
??	1024
machine	CAMIO72u
status	[0001] CAMIO Studio started

State	Action
CAMIO Studio started	power=ON controllermode=MANUAL execution=IDLE
CAMIO Studio stopped	power=OFF controllermode=UNAVAILABLE execution= UNAVAILABLE
DMIS Program started	power=ON controllermode=AUTOMATIC execution=EXECUTING
DMIS Program stopped	power=ON controllermode=MANUAL execution=IDLE
DMIS command error -	power=ON controllermode=MANUAL execution=IDLE error=message following dash
DMIS Program opened -	power=ON controllermode=MANUAL execution=IDLE program=message following dash
Side effects	RPM and xyz move if automatic and executing

DIRECTIONS TO CONFIGURE NIKON LOG FILE AGENT.

Modify Config.ini in C:\Program Files\MTConnect\MTConnectAgentNikonx64directory

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

```
[GLOBALS]
Config=NEW
```

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

```
MTConnectDevice=M1, M2, M3
```

3) 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$\logfolder\Events_log_BP_NIKON.txt
[M2]
ProductionLog=\\rufous\c$\logfolder\Events_log_BP_NIKON.txt
[M3]
ProductionLog=\\synchro\c$\logfolder\Events_log_BP_NIKON.txt
```

4) Start Nikon agent, the agent will detect a new configuration, and then write a new Devices.xml file to add the new devices.
5) If it works config.ini tag should say :"Config=UPDATED" if a problem tag will say: "Config=ERROR"

Source Code Explained

```
void AgentConfigurationEx::initialize(int aArgc, const char *aArgv[])
{
	std::string cfgfile = Globals.inifile; // "Config.ini"

	if(GetFileAttributesA(cfgfile.c_str())!= INVALID_FILE_ATTRIBUTES)
	{
		config.load( cfgfile );
		Globals.ServerName=config.GetSymbolValue("GLOBALS.ServiceName",
Globals.ServerName).c_str();
		MTConnectService::setName(Globals.ServerName);

		_devices = config.GetTokens("GLOBALS.MTConnectDevice", ",");

	}
```

Most importantly 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, HttpPort(default 5000), and ResetAtMidnite.`