

LoggerService2 User Guide

Date	Version Number	Document Changes
1/1/2020	1.0	N. Benson: Initial draft
11/23/2020	1.1	N. Benson; update screenshots / etc.
12/10/2020	1.1	N. Benson; added installation procedures

Table of Contents

1.	1.....	General Overview	4
2.	2.....	Installation	4
2.1.	Installing the LoggerService Deployment Bundle		4
2.1.1.	Get the bundle		4
2.1.2.	Prepare the bundle on the server.....		4
2.1.3.	Prepare Windows for the installer script.....		4
2.1.4.	Running the Installer Script.....		5
2.2.	Activating the License		8
2.2.1.	Manual Request for an Electronic License.....		8
2.2.2.	Installer Bundle Request for an Electronic License.....		10
2.2.3.	Install the License.....		10
3.	3.....	User Interface Overview	11
3.1.	Main Display.....		11
3.1.1.	Logger Service Settings		11
3.2.	Activity Selection.....		12
3.3.	Runtime/Status Information:.....		12
4.	4.....	Activity: Configuration	13
4.1.	Local WITSML Server.....		13
4.2.	Connections		14
4.2.1.	Connection Protocols.....		15
4.2.2.	Connection Types.....		17
4.2.3.	Outgoing Data		17
4.3.	Log Configuration.....		18
4.3.1.	Other Settings		18
5.	5.....	Activity: Monitoring	20
5.1.	Error Log Monitor		20

6.	6.....	Activity: Sync Config	
		21
7.	7.....	Runtime Information	
		21
8.	8.....	Basic Procedures	
		21
8.1.	Creating a New Well		21
8.2.	Creating a Sidetrack		21
9.	9.....	Troubleshooting	
		22

1. General Overview

DigiDrill LoggerService2 is multi-protocol, multi-media rig data multiplexor and aggregator. Data can be collected and re-transmitted across a variety of protocols and media (connection types). Data can also be stored locally as WITSML 1.4.1. This WITSML-formatted data can be synchronized to/from remote servers.

The application provides a variety of functionality for applying formulas, offsets, etc. to data being recorded. The WITSML server used to record aggregated data can be either local, or remote.

Additionally, there are specific features designed to support RTOC operations including auto-recording surveys, notifying of jumps in depth/time data indices, setting recording intervals, etc.

2. Installation

The LoggerService application has several dependencies that must be installed and running in order for it to operate properly.

- A local (PDS) WITSML server
- MongoDB for WITSML data storage
- CodeMeter runtime licensing service
- LoggerService application & UI

A complete application installation “bundle” can be provided which installs and configures each component and generates a license request that DigiDrill can authorize and apply to the server. The installation bundle supports

- Windows 8.1 / Server 2012
- Windows 10 / Server 2016+

2.1. Installing the LoggerService Deployment Bundle

2.1.1. Get the bundle

DigiDrill will provide you with an installation bundle ZIP. This will most likely be the URL required to download the latest version of the bundle.

2.1.2. Prepare the bundle on the server

Copy the bundle to the server, and uncompress it somewhere. Typically we recommend “C:\Temp\Bundle”. Documentation moving forward will reference this path.

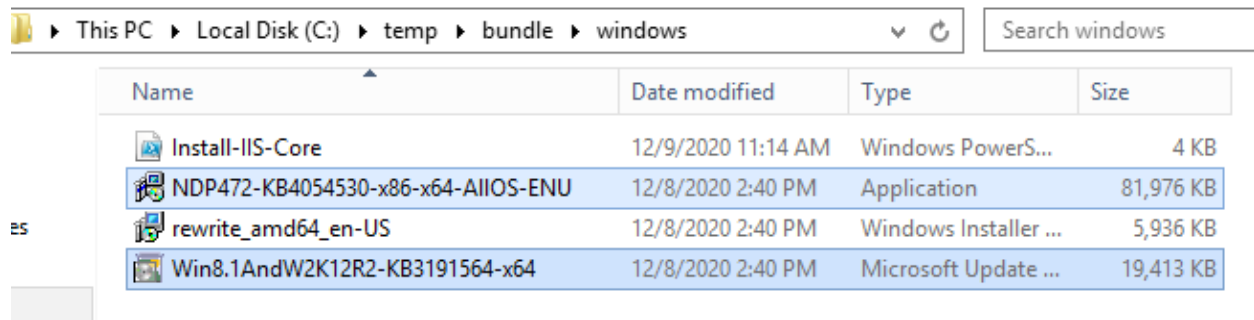
2.1.3. Prepare Windows for the installer script

2.1.3.1. Server 2012 Installation Requirements

In order to reduce the number of reboots required during installation, and keep the installation process easy we recommend that you install the following two updates **manually** before running the install script. Each update *may* require a reboot, depending on the state of the operating system. Some systems may have one or both of these updates already applied. This step is not required for Server 2016+.

The prerequisites can be found in “C:\Temp\Bundle\windows”. They should be installed manually in the following order:

- **Win8.1AndW2K12R2-KB3191564-x64.exe**
 - o This updates the system to the latest Windows Powershell, required by the installer scripts
- **NDP472-KB4054530-x86-x64-AIOS-ENU.exe**
 - o This is the .NET 4.7 runtime offline installer, required by the LoggerService application



Each of these installers may require a reboot when complete. Once each has been installed, and system rebooted the final time you are ready to move forward and execute the installer script.

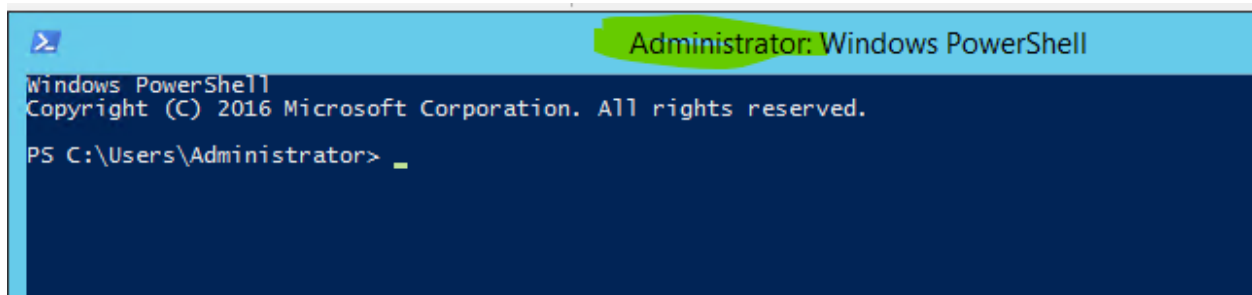
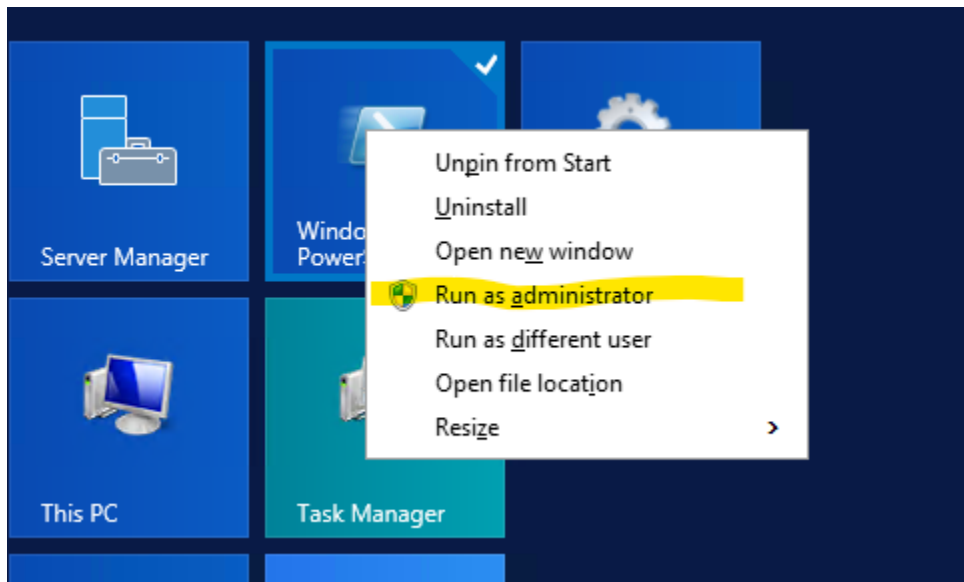
2.1.3.2. Server 2016 Installation Requirements

Windows 10 / Server 2016+ have all the required prerequisites installed by default. There are no additional steps beyond running the installer script.

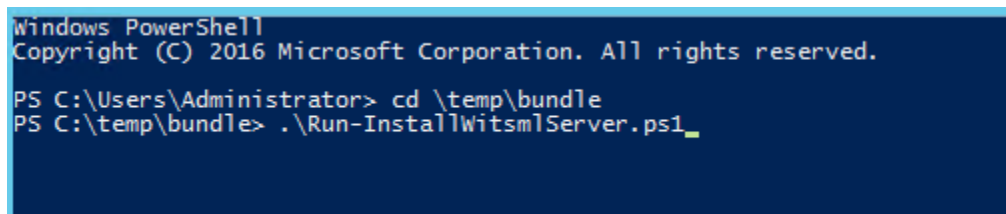
2.1.4. Running the Installer Script

The installer script is a collection of Powershell scripts which automate the installation of required Windows features, the WITSML server, DigiDrill LoggerService application, and the DigiDrill/PDS WITSML Studio Desktop application.

The first step is to open a Powershell prompt **as an administrator**.



Once you have opened the Powershell prompt, change directories to “C:\temp\bundle”, then execute the Run-InstallWitsmlServer.ps1 script.



You will immediately be prompted to enter information regarding the WITSML service user (for communicating with the WITSML server) as well as the MongoDB user/database/password (to allow the WITSML server to communicate with MongoDB).

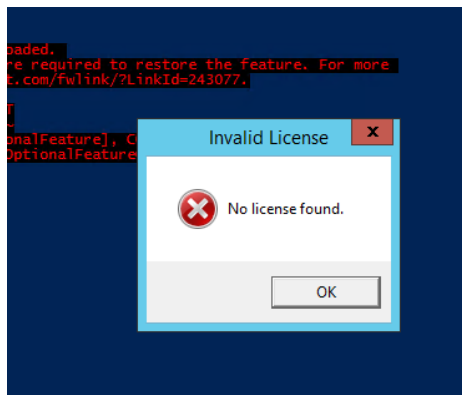
Property	Description	Notes
storeUser	The username provided to the LoggerService application to allow	

	it to connect, authenticate, and use the local WITSML server	
storePassword	The password to be used by the LoggerService application when communicating with the local WITSML server	
dbName	The name of the MongoDB database to be used by the local WITSML server	This should always be set to "WitsmlStore"
dbUser	The name of the MongoDB user allowing the WITSML server to communicate with MongoDB	
dbPassword	The password to be used by the WITSML server when communicating with MongoDB	

```
cmdlet Run-InstallWitsmlServer.ps1 at command pipeline position 1
Supply values for the following parameters:
storeUser: loggerservice
storePassword: dig1dr111
dbName: WitsmlStore
dbUser: witsmladmin
dbPassword: dig1dr111_
```

Once all the values have been entered, the script will proceed in installing and configuring the application all required services. This part of the process can take anywhere from 10-45 minutes depending on the speed of the server, and the status of the components installed. Specifically, if the script has to install any components required for the web server (IIS) this could take a while.

Note: if you are installing on a clean machine that has not previously had a license activated on it, you will receive the following message at the end of the installer:



Ignore this and click "OK". The script will generate a license request file to be sent to DigiDrill, which will in turn send back a license activation to be applied to the server.

The application is now installed.

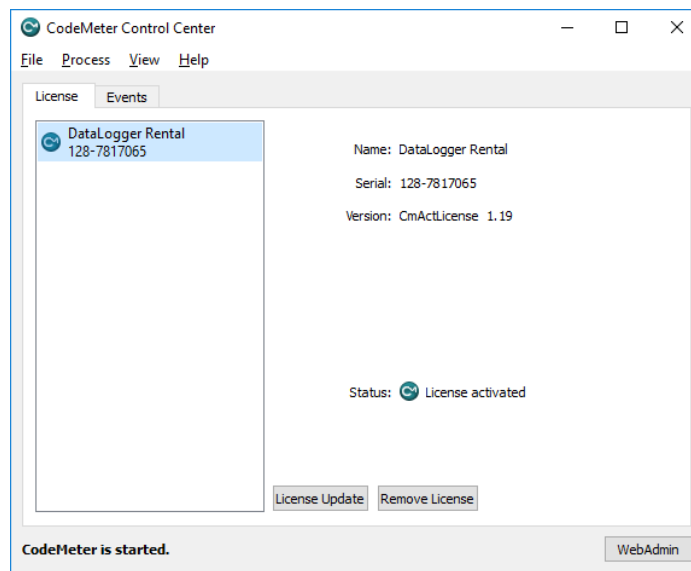
2.2. Activating the License

Once the installation script has completed, the license needs to be activated.

If you already have an activated license on the computer, you can skip this step.

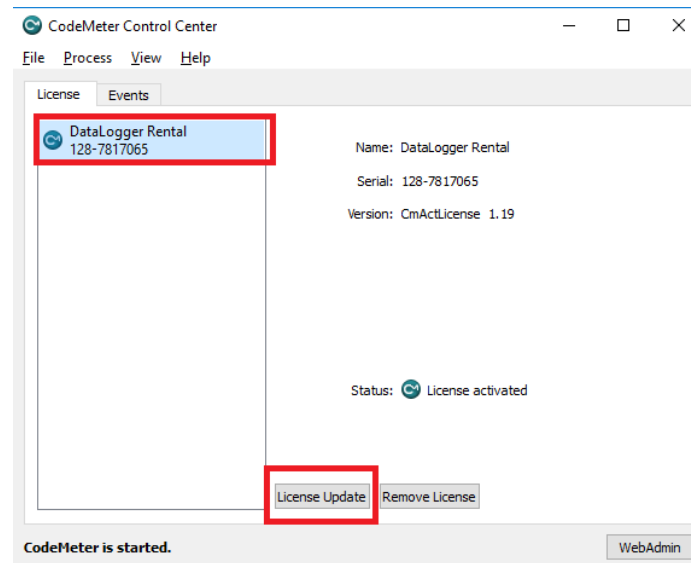
If you are installing from the installer bundle, skip to 2.2.2.

If you have received a license activation file from DigiDrill, skip directly to 2.2.3.

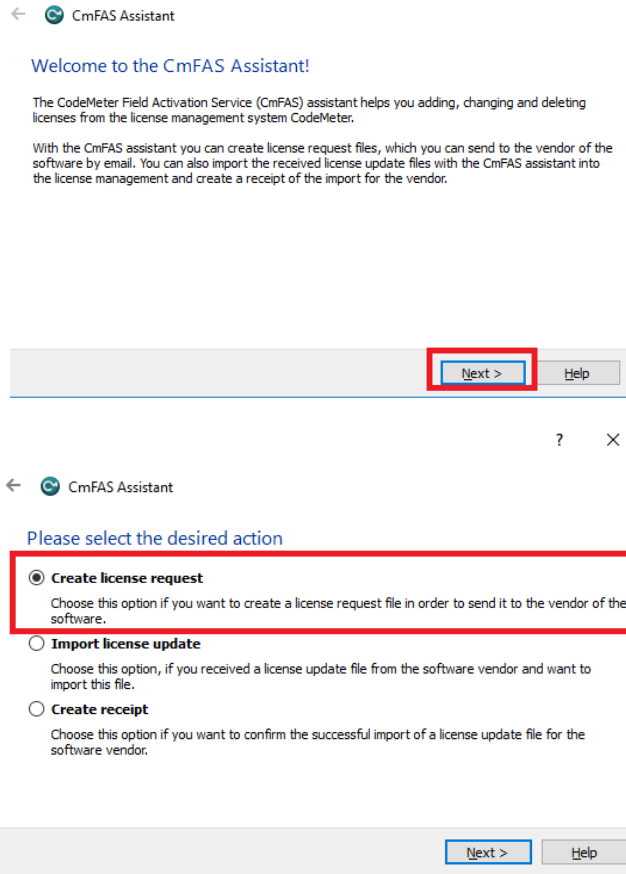


2.2.1. Manual Request for an Electronic License

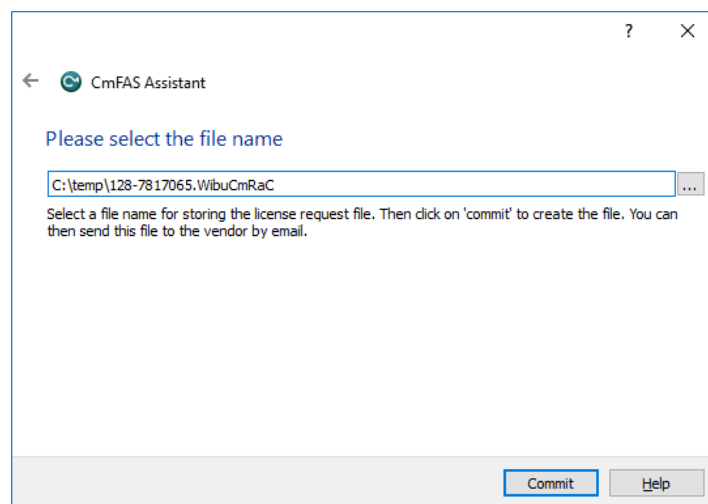
- Open the CodeMeter Control Center (Start Menu -> CodeMeter -> CodeMeter Control Center)
- Click on the "DataLogger Rental" entry in the license list
- Click on the "Activate License" or "License Update" button



- On the first page of the wizard click “Next”



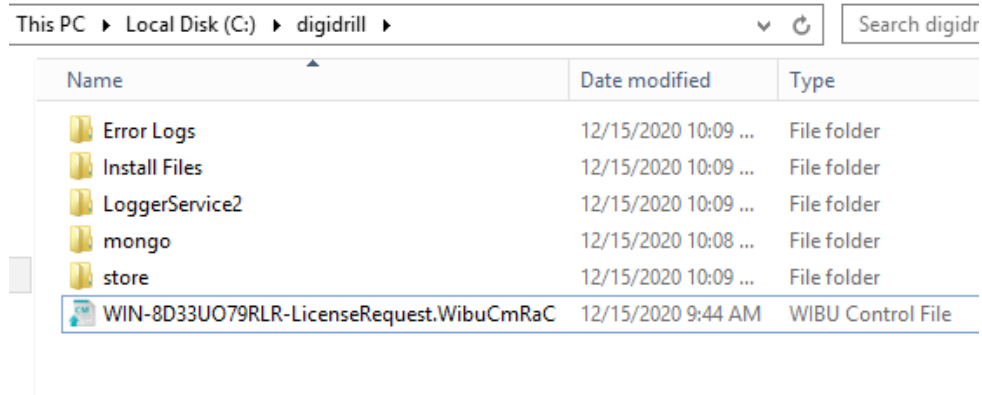
- Click “Next”
- When prompted, save the license request file to the computer



- Retrieve the license request file from the destination folder
- Send the request via e-mail to DigiDrill

2.2.2. Installer Bundle Request for an Electronic License

The installer bundle automatically creates a license update request file located in "C:\DigiDrill". The file will be named to match the "Computer Name" property of the server, with the extension "WibuCmRAC"

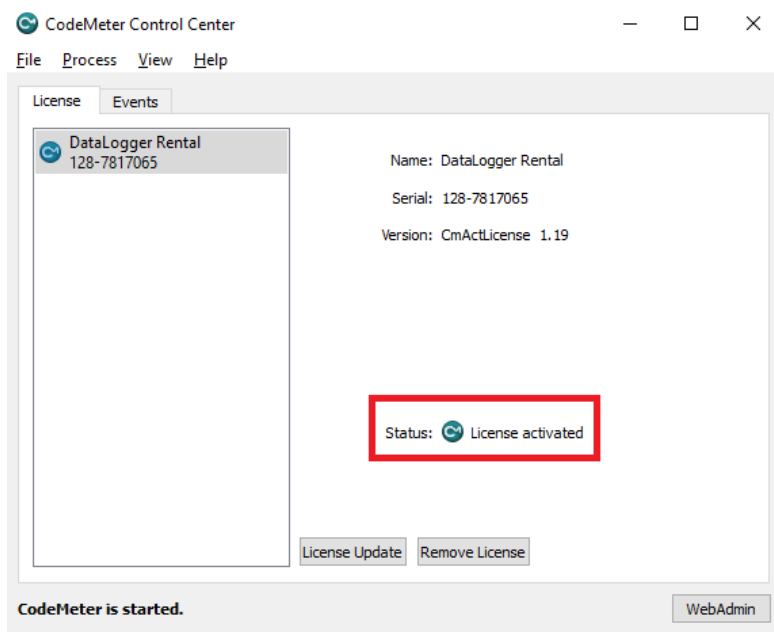


Send this file via e-mail to DigiDrill to activate it. Once you receive the activation file (WibuCmRAU) you can move on to 2.2.3.

2.2.3. Install the License

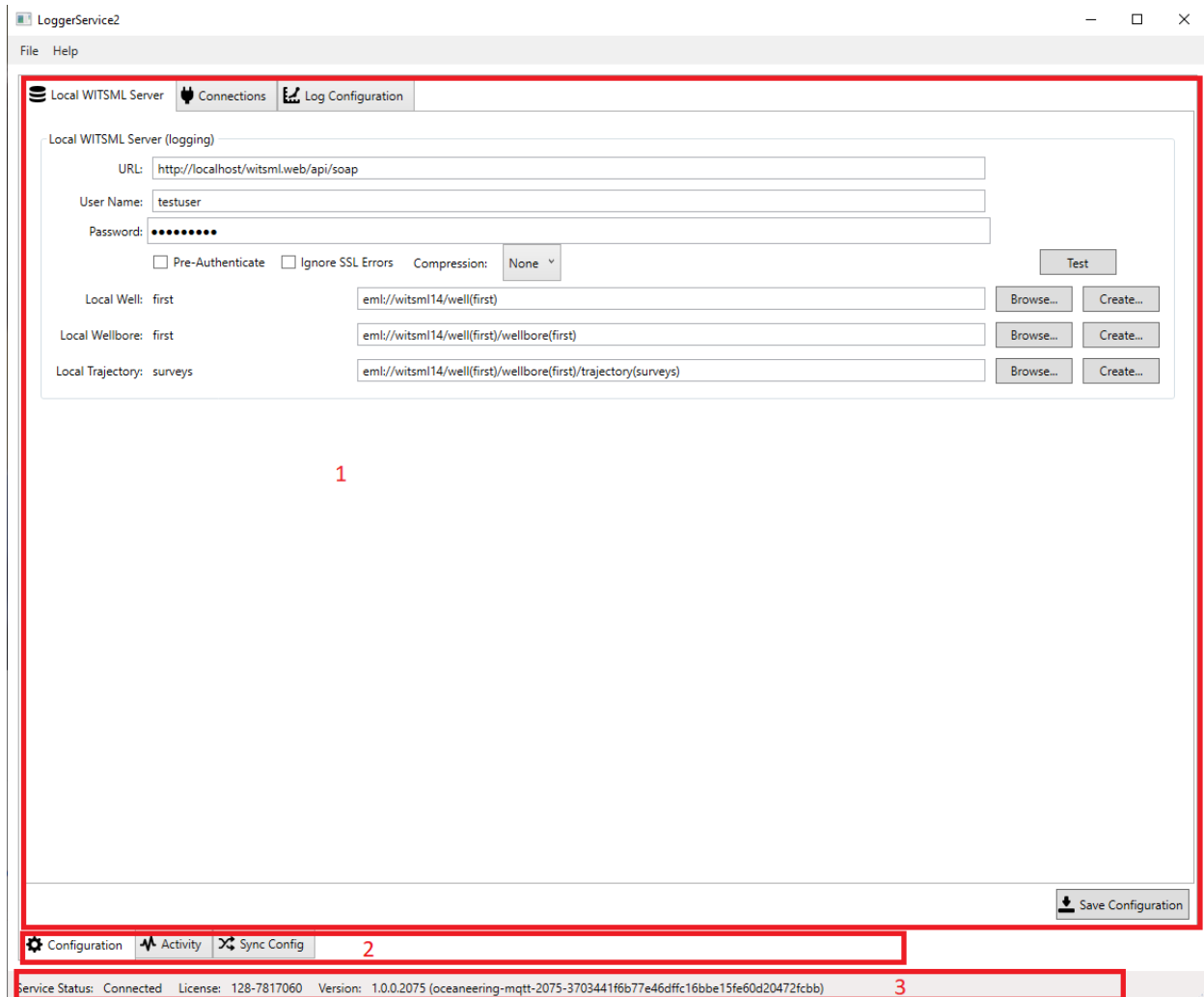
Once a license request is received DigiDrill can generate a new electronic license file. We will send back a CodeMeter "remote update" file that contains the license. It needs to be installed/imported onto the computer to complete the process.

- Receive the remote update file from DigiDrill (WibuCmRAU)
- Copy the file to the computer that generated the request
- Double click on the file and answer "yes" when prompted to import/install the license.
- Verify in the CodeMeter Control Panel that the license has been activated



3. User Interface Overview

The LoggerService has 3 distinct control sections for setting up a job, managing settings, and viewing activity. You can see these sections in the screenshot below. This document will cover the setup and use for LoggerService.

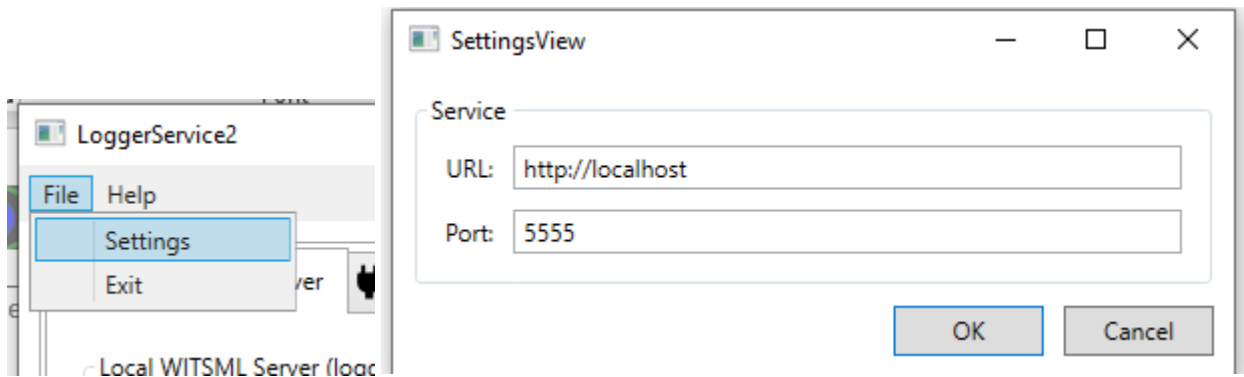


3.1. Main Display

Section 1 is the main display area, and is where the active activity is displayed.

3.1.1. Logger Service Settings

These settings should be ignored. They allow changing the communication URL and port that that administration interface uses.

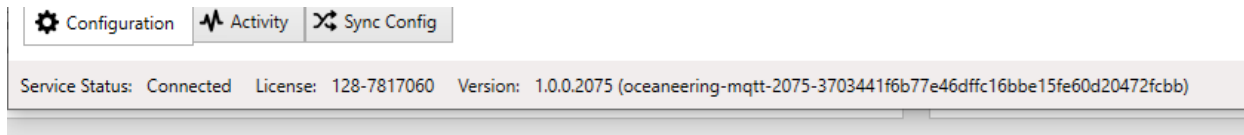


3.2. Activity Selection

Section 2 is where the user selects the activity (configuration, view data flow activity, configuration synchronization, etc.). The options and displays for the activity are shown in Section 1 – the main display.

3.3. Runtime/Status Information:

Section 3 is responsible for displaying the current runtime information as well as version/key numbers and current status.



4. Activity: Configuration

This activity allows the user to:

- Configure the local logging database (Local WITSML Server settings)
 - o Local server URL/credentials
 - o Local server communication settings
 - o Local server well/wellbore/trajectory for logging
- Create new incoming/outgoing connections
 - o WITS-0 (Serial/IP server or client)
 - o MQTT Client (IP client)
 - o Serial Delimited (Serial/IP server or client)
- Configure the properties for a connection
 - o Connection-specific properties and settings

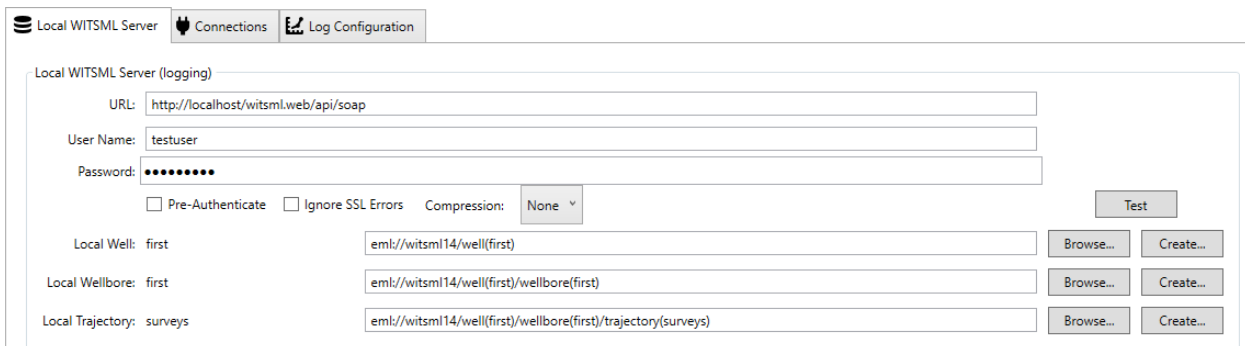
4.1. Local WITSML Server

The local WITSML server is the heart of all data storage for the LoggerService2 aggregator application. Data is collected from a variety of different connections, optionally has filters and formulas applied, and then is stored natively as WITSML 1.4.1 in the local store.

The configuration settings allow configuring the URL, authentication, connection options, and destination well/wellbore/trajectory.

Technically, the WITSML server is accessed via HTTP(S) and the WITSML SOAP API, so it can live anywhere network-accessible to the LoggerService. While it is recommended it live on the same hardware as the LoggerService it could be remote if required. “Local” is used primarily to denote that the LoggerService application expects to have **full control** of this server instance.

A properly configured LoggerService should have a URL, user, password. It must also have selections for the Local Well, Local Wellbore, and Local Trajectory.



Local WITSML Server (logging)

URL:

User Name:

Password:

Pre-Authenticate Ignore SSL Errors Compression:

Local Well: first

Local Wellbore: first

Local Trajectory: surveys

Property / Control	Required	Description
URL	YES	The URL of the WITSML server instance to be used by the LoggerService to record data
User Name	YES	The user name for the WITSML server instance
Password	YES	The password for the WITSML server instance
Pre-Authenticate	NO	Forces the connection to authenticate ALL requests to the web service, including those that may not technically require it. This option is recommended in most cases and causes no harm if enabled even when not required.
Ignore SSL Errors	NO	Allows the LoggerService to communicate via SSL to the WITSML server instance even when bad certificates are encountered (e.g. self-signed local certificates)
Compression	NO	Configures the WITSML compression type to be used. GZip is currently the only available compression method. This should only be used with servers that are known to support GZip and defaults to "None". This option is recommended when the destination server is known to support it. The PDS WITSML server provided with LoggerService does implement it.
Test		Tests the connection with the configured URL and credentials
Local Well	YES	Allows selection or creation of the WITSML well object that will contain all recorded data and objects
Local Wellbore	YES	Allows selection or creation of the WITSML wellbore object that will contain recorded log, mudlog objects and data
Local Trajectory	YES	Allows selection or creation of the WITSML trajectory object that will be used to store automatically-recorded surveys

4.2. Connections

Connections are uni-directional/bi-directional connections to/from data sources. Data can be accepted or transmitted to data sources (based on the connection type and protocol). For more information about specific connection protocols, see below.

4.2.1. Connection Protocols

4.2.1.1. *Wits*

Industry-standard WITS-0 data is supported over both serial (RS-232) and TCP/IP connections. In either case, a raw ASCII WITS-0 stream is expected, and data can be both received and transmitted over any kind of connection type.

For RS-232 (serial) communication, the LoggerService supports the two most common transmission speeds:

- 9600
- 115200

A null modem cable or other device (Moxa, etc) must be used to connect the computer systems. A regular serial cable will not work.

For TCP/IP connections, the LoggerService can be configured to act as either an IP client (LS connects to another computer/device that is publishing) or an IP Server (LS hosts a TCP/IP server that allows other devices connect and consume published data, or publish data to LS).

4.2.1.2. *Delimited*

Custom data sources that produce delimited ASCII data streams (configurable line endings and delimiters) are supported over both serial (RS-232) and TCP/IP connections. In either case, a raw ASCII delimited stream is expected, and data can be both received and transmitted over any kind of connection type.

For RS-232 (serial) communication, the LoggerService supports the two most common transmission speeds:

- 9600
- 115200

A null modem cable or other device (Moxa, etc) must be used to connect the computer systems. A regular serial cable will not work.

For TCP/IP connections, the LoggerService can be configured to act as either an IP client (LS connects to another computer/device that is publishing) or an IP Server (LS hosts a TCP/IP server that allows other devices connect and consume published data, or publish data to LS).

The delimited data source type expects a “CSV-like” data stream. Column headers are not expected or supported, but character-delimited “lines” or rows of data are. Both the delimiter character and the line ending are configurable. Additionally, the connection can be configured with the expected number of columns.

4.2.1.2.1 Delimited Configuration

Connections

Active	Name	Protocol	Type	Serial Port	Speed	IP Address	TCP Port
<input type="checkbox"/>	IPServer	Wits	IPServer		9600	127.0.0.1	5556
<input type="checkbox"/>	IPClient	Wits	IPClient			127.0.0.1	6666
<input checked="" type="checkbox"/>	MQTT Client	MqttJson	MqttClient			127.0.0.1	1883
<input checked="" type="checkbox"/>	Delimited	Delimited	Serial	COM1	9600		
<input type="checkbox"/>							

Extended Properties

Line Ending: LF

Column Delimiter: Tab

Expected Column Count: 8

Name	Description
Line Ending	The character combination to expect as the line/record terminator. Supports: LF, CRLF, CR
Column Delimiter	The value delimiter per line/record. Supports: Tab, Comma, Space
Expected Column Count	The number of columns to expect per line/record. Records with the incorrect number of values will be ignored and generate a warning in the error log.

4.2.1.3. MQTTJson

This connection type expects to receive JSON data from an MQTT broker. The extended configuration properties allow you to specify broker-specific settings.

4.2.1.3.1 MQTTJson Configuration

Connections

Active	Name	Protocol	Type	Serial Port	Speed	IP Address	TCP Port
<input type="checkbox"/>	IPServer	Wits	IPServer		9600	127.0.0.1	5556
<input type="checkbox"/>	IPClient	Wits	IPClient			127.0.0.1	6666
<input checked="" type="checkbox"/>	MQTT Client	MqttJson	MqttClient			127.0.0.1	1883
<input checked="" type="checkbox"/>	Delimited	Delimited	Serial	COM1	9600		
<input type="checkbox"/>							

Extended Properties

Server:

Port:

Auth. Enabled

User Name:

Password:

Topics:

group-1
group-2
group-3

Outgoing Data

Setting / Property	Description
Server	The IP address / host name of the MQTT broker
Port	The port of the MQTT broker
Auth. Enabled	Enables / Disables MQTT authentication
User Name	The MQTT user
Password	The MQTT password

Topics	The list of topics to subscribe to. Topics can be added or removed by selecting a topic / clicking remove, or typing a topic name and clicking add.
---------------	---

4.2.2. Connection Types

4.2.2.1. TCP Client (IPClient)

The TCP client connection type allows the LoggerService to connect to a remote TCP server to transmit or receive data. The remote server IP address and port can be configured. The TCP client connections will attempt to automatically reconnect if disconnected. If there is an error trying to connect, a message will be displayed in the error log monitor and the error log file. Communication is bi-directional.

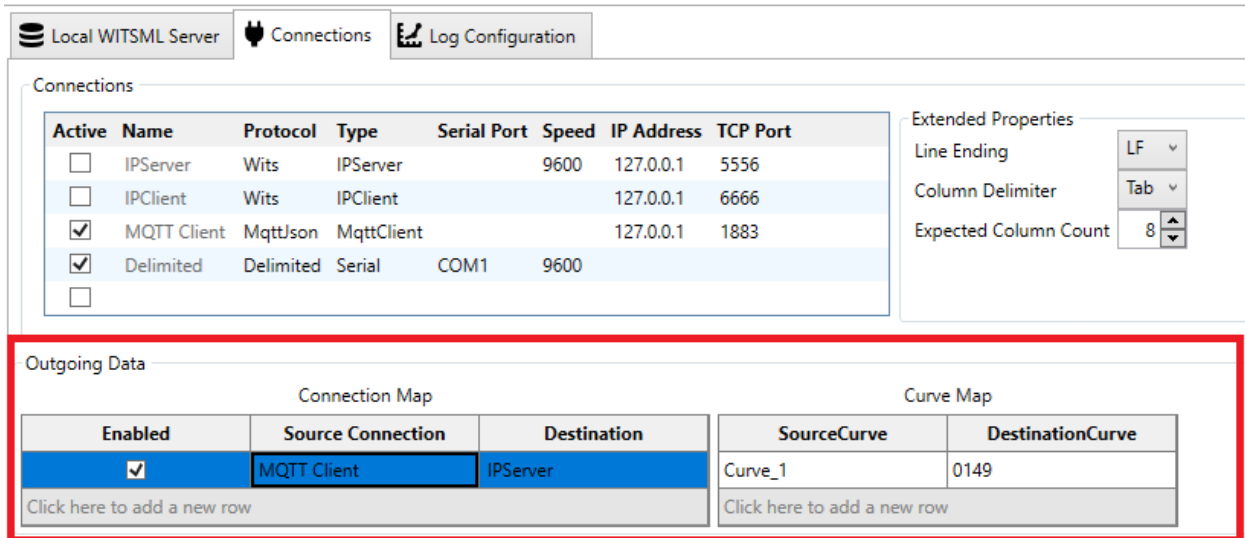
4.2.2.2. TCP Server (IPServer)

Implements a TCP server for hosting incoming connections. Communication is bi-directional.

4.2.2.3. MQTTClient

Implements support for connecting to an MQTT broker as a TCP/IP client with custom authentication and topic subscriptions. Communication is uni-directional (receive only).

4.2.3. Outgoing Data



The screenshot shows the 'Connections' tab in the LoggerService interface. It features a table of connections and an 'Extended Properties' panel.

Active	Name	Protocol	Type	Serial Port	Speed	IP Address	TCP Port
<input type="checkbox"/>	IPServer	Wits	IPServer		9600	127.0.0.1	5556
<input type="checkbox"/>	IPClient	Wits	IPClient			127.0.0.1	6666
<input checked="" type="checkbox"/>	MQTT Client	MqttJson	MqttClient			127.0.0.1	1883
<input checked="" type="checkbox"/>	Delimited	Delimited	Serial	COM1	9600		
<input type="checkbox"/>							

Extended Properties:

- Line Ending: LF
- Column Delimiter: Tab
- Expected Column Count: 8

The 'Outgoing Data' section is highlighted with a red box and contains two tables:

Connection Map		
Enabled	Source Connection	Destination
<input checked="" type="checkbox"/>	MQTT Client	IPServer
Click here to add a new row		

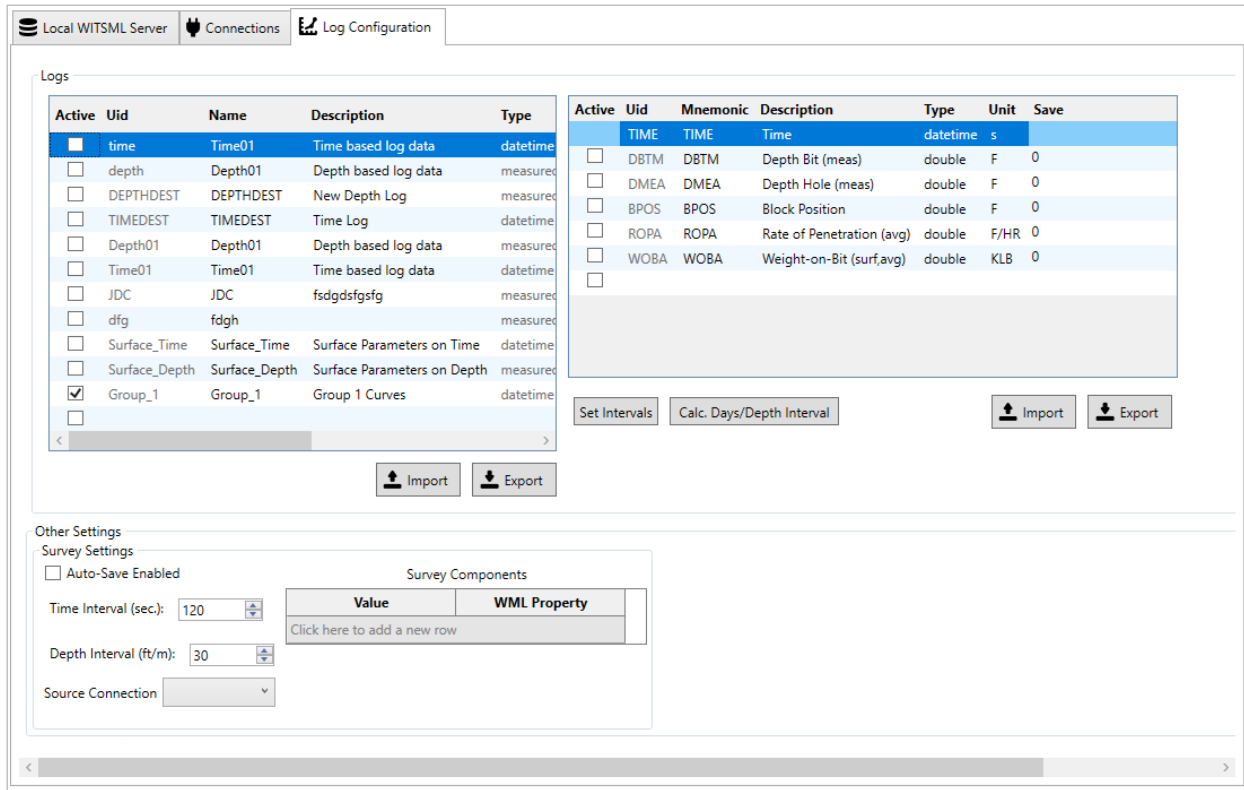
Curve Map	
SourceCurve	DestinationCurve
Curve_1	0149
Click here to add a new row	

The Outgoing Data sections allows the user to pick specific curves from specific sources, and rebroadcast them to other connections as different curves. This allows the application to act as a multiplexer by receiving and rebroadcasting data to/from various connections.

4.2.3.1. Connection Map

The connection map is a list of inbound -> outbound mappings. Each entry has both a source connection and a destination connect. Each entry also allows for specifying the specific curves to rebroadcast from the source to the destination.

4.3. Log Configuration



The screenshot displays the 'Log Configuration' window. At the top, there are three tabs: 'Local WITSML Server', 'Connections', and 'Log Configuration'. The 'Log Configuration' tab is active and shows a 'Logs' section with a table of log entries. Below this is an 'Other Settings' section with 'Survey Settings' and 'Survey Components'.

Active	Uid	Name	Description	Type
<input checked="" type="checkbox"/>	time	Time01	Time based log data	datetime
<input type="checkbox"/>	depth	Depth01	Depth based log data	measured
<input type="checkbox"/>	DEPTHDEST	DEPTHDEST	New Depth Log	measured
<input type="checkbox"/>	TIMEDEST	TIMEDEST	Time Log	datetime
<input type="checkbox"/>	Depth01	Depth01	Depth based log data	measured
<input type="checkbox"/>	Time01	Time01	Time based log data	datetime
<input type="checkbox"/>	JDC	JDC	fsdgsfsgfsg	measured
<input type="checkbox"/>	dfg	dfg	dfg	measured
<input type="checkbox"/>	Surface_Time	Surface_Time	Surface Parameters on Time	datetime
<input type="checkbox"/>	Surface_Depth	Surface_Depth	Surface Parameters on Depth	measured
<input checked="" type="checkbox"/>	Group_1	Group_1	Group 1 Curves	datetime

Active	Uid	Mnemonic	Description	Type	Unit	Save
<input type="checkbox"/>	TIME	TIME	Time	datetime	s	
<input type="checkbox"/>	DBTM	DBTM	Depth Bit (meas)	double	F	0
<input type="checkbox"/>	DMEA	DMEA	Depth Hole (meas)	double	F	0
<input type="checkbox"/>	BPOS	BPOS	Block Position	double	F	0
<input type="checkbox"/>	ROPA	ROPA	Rate of Penetration (avg)	double	F/HR	0
<input type="checkbox"/>	WOBA	WOBA	Weight-on-Bit (surf,avg)	double	KLB	0

Buttons: Set Intervals, Calc. Days/Depth Interval, Import, Export

This region controls the configuration of the recorded data. It allows the user to configure multiple logs with different index types (measured depth, date/time, etc.) as well as details about the curves to be recorded, the source of the data for the curves, and any translation/transformation formulas that need to be applied.

4.3.1. Other Settings

Automatic survey recording provides a way to translate WITS-0 07xx records into directional surveys.

For this process to work, there are three things that need to be configured

- Enable Auto-Save
- The depth interval for recording
- The time interval for recording
- The connection to watch for survey components

- The mapping of incoming survey component curves to WITSML trajectoryStation (survey) properties

Other Settings

Survey Settings

Auto-Save Enabled

Time Interval (sec.):

Depth Interval (ft/m):

Source Connection:

Survey Components

Value	WML Property
Source	Name
Source	Name
Source	Name
Click here to add a new row	

4.3.1.1. Survey Depth / Time Intervals

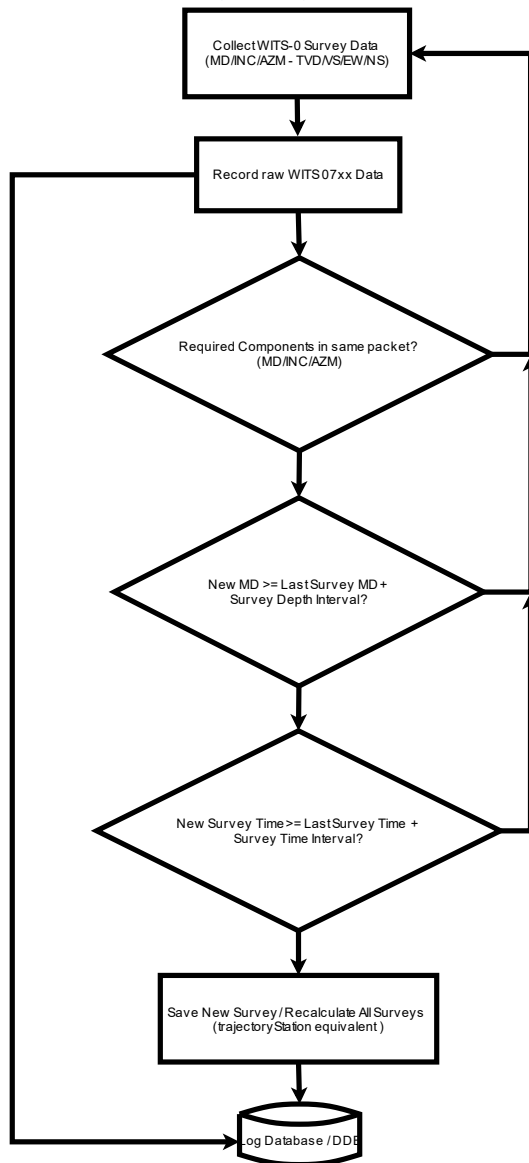
The survey depth and time intervals record how frequently full surveys get generated from WITS-0 07xx records.

The depth interval controls the minimum distance between surveys in measured depth, and the time interval controls how frequently we will record those surveys if the depth interval has been met.

These two limits are always applied. If the time between survey packets has exceeded the time interval, but the depth interval has not been met then a survey will not be recorded. If the depth interval has been met, but the time interval has not, a survey will not be recorded.

If both limits has been met, and we received a full packet of information (MD/INC/AZM) then a new survey will be recorded to the log database, and the full list of surveys will be recalculated.

4.3.1.2. Streaming Survey Process Flow Chart



5. Activity: Monitoring

5.1. Error Log Monitor

This tab allows the user to monitor diagnostic information produced by the LoggerService. This includes debug, informational, warning, and error messages. The default setting for log messages displayed here is DEBUG, which shows all levels of information.

Well Information LiveLog/WITSML Server Settings **LoggerService Settings** Error Log Monitor Log Record Selection Depth Mappings Time to Depth Mappings

Refresh every seconds. Level: Log File Level:

```
2018-09-20T10:12:23: ERROR: Loaded WITS fields.
2018-09-20T10:12:23: ERROR: Loaded table-based curves
2018-09-20T10:12:24: ERROR: Loaded table columns
```

6. Activity: Sync Config

7. Runtime Information

This displays the database in use, the version of the software, the license serial number, and the status of the service (running/stopped/restarting).

8. Basic Procedures

8.1. Creating a New Well

Creating a new well involves three steps:

- 1) Create a new log database (.DDB file)
- 2) Update the well header information
- 3) Apply any configuration settings required (connections, mappings, etc)

8.2. Creating a Sidetrack

The current process for creating a sidetrack is

- 1) Create a new log database (DDB)
- 2) Apply header information and other configuration settings (connections, etc)
- 3) Save changes
- 4) Allow the database to upload to the LiveLog server
- 5) Add pds@digidrill.com to the user list for the new well
- 6) DigiDrill will then copy the data required to start the sidetrack
- 7) PDS will configure StoreSync to transfer to the NDR
- 8) Add any other users and notify them of the new sidetrack UIDs

