

Dataframe Utilities Extension

User Guide

Version 1.2.5

**Copyright © 2016 PTC Inc. and/or Its Subsidiary Companies. All Rights Reserved.**

User and training guides and related documentation from PTC Inc. and its subsidiary companies (collectively “PTC”) are subject to the copyright laws of the United States and other countries and are provided under a license agreement that restricts copying, disclosure, and use of such documentation. PTC hereby grants to the licensed software user the right to make copies in printed form of this documentation if provided on software media, but only for internal/personal use and in accordance with the license agreement under which the applicable software is licensed. Any copy made shall include the PTC copyright notice and any other proprietary notice provided by PTC. Training materials may not be copied without the express written consent of PTC. This documentation may not be disclosed, transferred, modified, or reduced to any form, including electronic media, or transmitted or made publicly available by any means without the prior written consent of PTC and no authorization is granted to make copies for such purposes.

Information described herein is furnished for general information only, is subject to change without notice, and should not be construed as a warranty or commitment by PTC. PTC assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

The software described in this document is provided under written license agreement, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. It may not be copied or distributed in any form or medium, disclosed to third parties, or used in any manner not provided for in the software licenses agreement except with written prior approval from PTC.

UNAUTHORIZED USE OF SOFTWARE OR ITS DOCUMENTATION CAN RESULT IN CIVIL DAMAGES AND CRIMINAL PROSECUTION. PTC regards software piracy as the crime it is, and we view offenders accordingly. We do not tolerate the piracy of PTC software products, and we pursue (both civilly and criminally) those who do so using all legal means available, including public and private surveillance resources. As part of these efforts, PTC uses data monitoring and scouring technologies to obtain and transmit data on users of illegal copies of our software. This data collection is not performed on users of legally licensed software from PTC and its authorized distributors. If you are using an illegal copy of our software and do not consent to the collection and transmission of such data (including to the United States), cease using the illegal version, and contact PTC to obtain a legally licensed copy.

**Important Copyright, Trademark, Patent, and Licensing Information:** See the About Box, or copyright notice, of your PTC software.

**UNITED STATES GOVERNMENT RESTRICTED RIGHTS LEGEND**

This document and the software described herein are Commercial Computer Documentation and Software, pursuant to FAR 12.212(a)-(b) (OCT’95) or DFARS 227.7202-1(a) and 227.7202-3(a) (JUN’95), and are provided to the US Government under a limited commercial license only. For procurements predating the above clauses, use, duplication, or disclosure by the Government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 (OCT’88) or Commercial Computer Software-Restricted Rights at FAR 52.227-19(c)(1)-(2) (JUN’87), as applicable. 01012015

**PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA**



[Software Change Log 3](#_Toc462417794)

[Introduction and Installation 3](#_Toc462417795)

[About the Dataframe Utilities Extension 3](#_Toc462417796)

[Installing the Dataframe Utilities Extension 3](#_Toc462417797)

[Configuration and Usage 5](#_Toc462417798)

[Usage 5](#_Toc462417799)

[Compatibility 7](#_Toc462417800)

# Software Change Log

|  |  |  |
| --- | --- | --- |
| Version | Release Date | Changes |
| 1.0 | 01/18/2016 | Initial Release |
| 1.2 | 09/26/2016 | Added blob capabilities. Extended the infotable format properties. |
| 1.2.5 | 14/02/2018 | Fixed an issue where signed values were not correctly parsed. This means that now you must always specify if a field is unsigned. |

# Introduction and Installation

Extensibility is a core aspect of the architecture and design of ThingWorx. Partners, third parties, and ThingWorx users can easily add new functionality into the system in a seamless manner. Extensions can be Service (function/method) libraries, Connector Templates, Widgets, and more.

This document provides installation and usage instructions for the Dataframe Utilities Extension.

# About the Dataframe Utilities Extension

The Dataframe Utilities Extension for the ThingWorx Platform allows encode and decode with complex frames of data in order to communicate with low level devices parse binary files and protocols. So, this extension allows you to skip the complex part of doing bit parsing and in order to obtain all the fields data inside a binary frame, and instead, you just need to create an infortable that describes the binary frame format.

You can use it in scenarios where you need to decode/encode data for embedded devices, or to ease the translation to and from a binary protocol, as well as using binary files within Thingworx.

## Installing the Dataframe Utilities Extension

|  |  |
| --- | --- |
| 1. From a web browser, launch ThingWorx. 2. Log into ThingWorx as an administrator. |  |
| 1. Go to **Import/Export > Import**. |  |
| 1. Click Choose File and select [DataframeUtilitiesExtension].zip 2. Click **Import**.  Note: If an **Import Successful** message does not display, contact your ThingWorx System Administrator. | Note: |
| 1. Click **Yes** to refresh Composer after importing the final extension. |  |
| 1. Confirm that the Extension has been imported properly. Check the Application Log for potential problems. |  |
|  |  |

# Configuration and Usage

In order to you use the Dataframe Utilities Extension, you must examine the firstly understand the configuration using a **FrameFormatDataShape.** Inside this datashape you can define the properties for each of the fields of the frame. Using this datashape you will create an infotable (either using the standard UI or using a script) that describes the frame that you need to encode or decode.

After you have an infotable describing the frame, you use the **FrameUtilitiesThing** thing which provides the following services:

* **DecodeBlob**: uses a frame format Infotable and a byte array (BLOB), returning a JSON with the decoded data. The result JSON contains the value for each of the fields.
* **DecodeHex**: uses a frame format Infotable and a data string returning a JSON with the decoded data. The data string represents a string with data in hexadecimal format. The result JSON contains the value for each of the fields.
* **EncodeBlob**: uses a frame format Infotable, with all the properties set and returns a byte array (BLOB) with the encoded data.
* **EncodeHex**: uses a frame format Infotable, with all the properties set and returns a string with data in hexadecimal format.

## Usage

At the base of this extension is the **FrameFormatDataShape** datashape. Using infotables with this datashape you can map the fields of the frame.



Figure FrameFormatDataShape fields

For example, if you have a frame from devices described as the following table:

Table Sample frame definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **BITS** | **0-7 (byte0)** | **8 (byte1, bit7)** | **9-15 (byte1, bit6-0)** | **16-31 (bytes 2, 3)** | **32-63 (bytes 4,5,6,7)** | **64-79** |
| **FIELD** | CODE, unsigned | Conf Status | Frame counter, unsigned | Device ID (LSB, unsigned) | Sensor Value (float32) | Sensor  Description |

The infotable that describes the preceding table is presented in the following figure. You can see that a row is used for each of the fields in the frame.

The extension supports fields of data that span across multiple bytes, with a portion of the bits in one byte, and another portion in another byte.

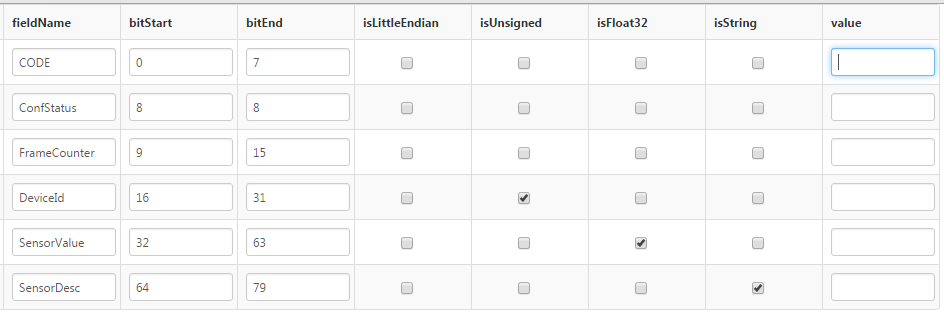


Figure Sample FrameFormatDatashape

The **FrameFormatDataShape** has the following fields:

* **fieldName:** The name of the field. This name is also used to construct the result JSON for frame decoding.
* **bitStart:** The bit where this frame starts. Please note that **bitStart** is included in the field (it’s a closed interval, where endpoints are included)
* **bitEnd:** The bit where this frame ends. Please note that **bitEnd** is included in the field (it’s a closed interval, where endpoints are included)
* **isLittleEndian:** For fields spanning across multiple bytes, specifies if the field is little endian (LSB first).
* **isUnsigned:** The value should be treated as unsigned.
* **isFloat32:** Value is an IEE754 single precision floating point. This starts at **bitStart** and ends at (**bitStart + 31**), so **bitEnd** is not used. **BitStart** must represent the start of a byte (divisible by 8)
* **isString:** Value represents a string, where each byte represents a character point. **BitStart** must represent the start of a byte (divisible by 8)
* **value:** The value for this field. Used for frame encoding.

The **FrameUtilitiesThing** has the following services:

* **DecodeBlob**: Decodes a data frame, using a given specification
  + **Inputs**:
    - **frameFormat**: An infotable with the **FrameFormatDataShape** describing the frame
    - **data**: The data to decode represented using a Byte Array (BLOB).
    - **offset** (optional, default 0): What is the byte offset the decoding should start from
  + **Result**: JSON data. For each of the fields in the input frame format, a corresponding property exists in the JSON, with the value equal to the computed value from the data
* **DecodeHex**: Decodes a data frame, using a given specification
  + **Inputs**:
    - **frameFormat**: An infotable with the **FrameFormatDataShape** describing the frame
    - **data**: The data to decode, represented as a string with hexadecimal data, where first byte (byte 0) represent the first two characters.
  + **Result**: JSON data. For each of the fields in the input frame format, a corresponding property exists in the JSON, with the value equal to the computed value from the data
* **EncodeBlob**:
  + **Inputs**:
    - **frameFormat**: An infotable with the **FrameFormatDataShape** describing the frame. The value field in this infotable must be filled.
  + **Result:** The encoded data, represented as Byte Array (BLOB)
* **EncodeHex**:
  + **Inputs**:
    - **frameFormat**: An infotable with the **FrameFormatDataShape** describing the frame. The value field in this infotable must be filled.
  + **Result:** The encoded data, represented as a string with hexadecimal data, where first byte (byte 0) represent the first two characters.

## Notes

There are two ways of using this extension. If the frame has a static format (each of the fields has a static location within the frame), then you can statically create an infotable, just as displayed in the example above.

If you are dealing with a dynamic frame format, that the FrameFormatDataShape infotable can be dynamically computed, based on custom business logic inside a service.

### Translating a javascript byte array into a BLOB

Because of Thingworx limitations, you cannot pass a javascript byte array into an service that requires a BLOB input.

However, you can use the following script to do it:

// having a javascript byte array does not translate directly to a BLOB result

// a workaround is to convert the byte array into a base64 string, and that can be fed to the service with a BLOB input parameter

for(var i=0;i<bytes.length;i++) {

// make sure this are bytes

bytes[i] &= 0xff;

// this line seems wierd, but it's because the byte Java type is from -127 to 127.

// our byte array is 0 to 255. So we shift to that domain.

bytes[i] = (bytes[i] > 127) ? (bytes[i] - 256) : bytes[i];

}

var result = base64EncodeBytes(

{

array: bytes

}

);

# Compatibility

This extension was tested for compatibility with the following ThingWorx Platform version(s) and Operating System(s). Please note that some model formats are not compatible with certain browsers.

|  |  |
| --- | --- |
| ThingWorx Platform Version | ThingWorx 6.5.0 – 8.2.1 |
| OS | Windows 7, Service Pack 1, iOS 10, Android 5, OSX 10.12 |
| Browser | Chrome 53, Firefox 48, |