

**Innomatix Remote DataStore**

**API Specification**

|  |  |
| --- | --- |
| Revision | 1.1 |
| Revision Date | 6-Sep-2017 |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Change** |
| 1.0 | 8-Jun-2015 | JLASH | Initial Creation |
| 1.1 | 6-Sep-2017 | JLASH | Minor updates re: communications interfaces |
|  |  |  |  |

Table of Contents

1.0 Overview 2

2.0 Scope 2

3.0 Server 2

4.0 Client 3

5.0 Remote DataStore API 3

5.1. DataStoreInit 3

5.2. DataStoreClose 3

5.3. GetDataBinInfo 3

5.4. Get Functions 4

5.5. Put Functions 5

5.6. API Return Values 6

1. Overview

This document defines the Remote DataStore API of the Innomatix Telematics Framework. The goal of this functionality is to provide a robust, fast and simple method for external applications to access information in the DataStore of a TF application.

The primary use of this functionality is to for the DAP Coprocessor to exchange data with the main DAP application. This is necessary to achieve an integrated platform for end-users to program with their own logic/algorithms.

1. Scope

The resource and functional limitations of a microcontroller have been considered during the development of this functionality. In particular, the following limitations were assumed to ensure the best possible performance:

* It is assumed that the coprocessor does not have an operating system and thus does not have threads. This reduces resources but does mean access to the Remote DataStore is done “as-needed” rather than done “in the background”.
* It is assumed that memory is limited (possibly as little as 8KB).
* Dynamic allocation of memory is kept to a bare minimum and is only performed during library initialization.

1. Server

The DataStore Server is a Telematics Framework module that must be included in a DAP project to provide the following high-level functionality for clients:

* GetDefinition – allows the client to retrieve data bin information such as the id, data type and the maximum data size (primarily for string/binary data bins).
* GetData – allows a client to retrieve data from a data bin.
* PutData – allows a client to update a data bin value.

1. Client

The client is the DAP Coprocessor. It connects to the Remote DataStore server and provides a local API for getting information about DataBins, for getting data from and publishing data to the Remote DataStore.

1. Remote DataStore API

The client API is in the C programming language. The overriding goal is simple and clean.

The API consists of the following public functions:

* 1. DataStoreInit

This function initializes the API and attempts to bring the communications interface up in order to connect to the Remote DataStore server.

results\_e DataStoreInit()

Return Values:

* Success – a connection was established
* NoConnection – the timeout expired without a connection
  1. DataStoreClose

This function disconnects from Remote DataStore server and brings down the communications interface. Other API function calls fail until a successful connection is again established.

* 1. GetDataBinInfo

Allow the client application to retrieve information about a data bin.

results\_e GetDataBinInfo( char \*Name, BinInfoStruct\_t &Info )

Return Values:

* Success – the BinInfoStruct has been filled with information about the requested data bin
* NoConnection – the connection has been lost, or the timeout expired with no response from the server
* InvalidBin – a data bin with the given name does not exist
* GeneralError – an unknown or unspecified error has occurred

BinInfoStruct is defined as

typedef struct

{

uint16\_t BinId;

DataType BinDataType;

uint8\_t BinName[ 64 ]; /\* NULL terminated \*/

}BinInfoStruct\_t

See RemoteDataStoreAPI.h for additional return values and structure fields.

The client provides a data bin name (string) and a structure pointer. The function retrieves information and populates the structure. GetDataBinInfo must be called before a data bin can be used as the BinInfoStruct contains the *BinId* that is used for Put and Get calls.

GetDataBinInfo should be called after each time a connection is established to ensure that any changes to the Remote DataStore are detected.

* 1. Get Functions

Get functions allow the client application to retrieve values from the Remote DataStore.

results\_e GetSignedInteger( uint32\_t BinId,

int32\_t \*Value, uint32\_t \*Timestamp )

results\_e GetUnsignedInteger( uint32\_t BinId,

uint32\_t \*Value, uint32\_t \*Timestamp )

results\_e GetDouble(uint32\_t BinId,

double \*Value, uint32\_t \*Timestamp )

results\_e GetString( uint32\_t BinId, char \*Str,

uint16\_t MaxStrLen, uint32\_t \*Timestamp )

results\_e GetBlob( uint32\_t BinId, uint8\_t \*Blob, uint16\_t MaxBlobLen,

uint16\_t \*ActualBlobLen, uint32\_t \*Timestamp )

Return Values:

* Success – the Value and Timestamp parameters have been filled
* NoConnection – the connection has been lost, or the timeout expired with no response from the server
* InvalidBin – a data bin with the given ID does not exist
* IncompatibleType – the specified data bin is not of the requested type and the data bin value cannot be convereted to the requested type.
* NoData – the specified data bin has not been written to yet
* BadParameter – the value buffer pointer is NULL
* TooBig – the value buffer is not big enough to hold the retrieved data, the value buffer is left unchanged
* InUse – the requested data bin is in use by another application or client and cannot be read
* GeneralError – an unknown or unspecified error has occurred

See RemoteDataStoreAPI.h for details

The client should always check the results before using the value.

The callers value is unchanged if the request is not successful.

The timestamp field of all Get functions is the timestamp that the requested data bin was last updated. NOTE as with the BinInfoStruct, the timestamp is relative to when the Remote DataStore was started and does not correlate to “clock time”.

The timestamp pointer may be NULL if the timestamp is not needed.

* 1. Put Functions

Put functions allow the client application to update a Remote DataBin value.

results\_e PutSignedInteger( uint32\_t BinId, int32\_t Value )

results\_e PutUnsignedInteger( uint32\_t BinId, uint32\_t Value )

results\_e PutDouble( uint32\_t BinId, double Value )

results\_e PutString( uint32\_t BinId, char \*Str )

results\_e PutBlob( uint32\_t BinId, unsigned char \*Blob,

uint16\_t BlobLen )

Return Values:

* Success – the data bin was successfully updated
* NoConnection – the connection has been lost, or the timeout expired with no response from the server
* InvalidBin – a data bin with the given ID does not exist
* IncompatibleType – the specified data bin is not of the requested type and the provided value cannot be converted to the proper type for the data bin.
* BadParameter – the value buffer pointer is NULL (for string and blob functions only)
* TooBig – the specified data bin is not big enough to hold the provided data, the data bin is left unchanged
* InUse – the requested data bin is in use by another application or client and cannot be updated
* ReadOnly – the requested data bin is read only and cannot be updated
* GeneralError – an unknown or unspecified error has occurred

See RemoteDataStoreAPI.h for details

The client should always check the results to ensure success.

* 1. API Return Values

Specific values for the Remote DataStore API function *results\_e* are as follows.

|  |  |
| --- | --- |
| Name | Value |
| eSuccess | 0x00 |
| eNoConnection | 0x01 |
| eNotAllowed | 0x02 |
| eServerFull | 0x03 |
| eInvalidBin | 0x04 |
| eNoData | 0x05 |
| eTooBig | 0x06 |
| eBadParameter | 0x07 |
| eIncompatibleType | 0x08 |
| eInUse | 0x09 |
| eReadOnly | 0x0A |
| eGeneralError | 0xFF |

See RemoteDataStoreAPI.h for details