The DataSync Driver

The DataSync driver provides a simple text-based API to the North device's Essential Data, allowing your application to synchronise with and adjust information from the database. Available for Commander and ObSys.

This document relates to DataSync driver version 2.0 and 2.1

Please read the *Commander Manual* or *ObSys Manual* alongside this document, available from *www.northbt.com*

Contents

Purpose of DataSync Driver

The DataSync driver provides a simple text-based API to the North device's Essential Data and Extra Data, allowing your application to synchronise with and adjust information from the database.

Use this driver to integrate data collected using North interface technology directly into your own application or product, such as a home automation controller.

The DataSync API uses a client-server model. The server provides values from a database to a single client device when requested or on a change of value. The client can request all values from the server or set a value.

Typically, your application would implement the client role, with the DataSync driver implementing the server role to provide values from the database.

The DataSync driver can also perform the client role instead – use this to connect two DataSync drivers on different North devices together and synchronise the Essential Data and Extra Data values between them.

Connect devices implementing the DataSync API using an RS232 or TCP/IP connection.

Other APIs are available: JSONData and JSONNotify drivers provide JSON web services; the Telnet driver provides a TCP query-response interface; the SG driver provides an RS232 request-reply interface; and many other standards-based drivers are available.

Values

You can connect to the DataSync API using simple text-based line commands. The API presents values from the North device's Essential Data and Extra Data. Essential Data contains 640 values on Commander, and 1280 values on ObSys. If necessary, start the Extra Data driver (which requires an interface licence) for an additional 1024 values. Access to these values can be controlled by configuring privilege levels within the driver.

The functions currently available from the API are:

- Value functions – list values from the database and adjust them
- Object functions – list object types from the database

Prerequisites

The Essential Data and/or Extra Data module should be configured, with access security levels set if required.

Detailed Operation

The DataSync API provides a simple text-based protocol for an application to remain synchronised with the data held in a North device's database. This database, created using Essential Data and Extra Data, is updated with data from the building – this will be site specific, but could include information from systems such as air conditioning, building automation, fire detection, HVAC, lighting, energy metering, security, etc.

The API uses a stateless client-server model. Typically, your application would implement the client role, with the North device implementing the server role to provide values from Essential Data and Extra Data.

The North device, or server, maintains the database and automatically sends notifications to the connected client device when a value changes. It will also respond to queries sent by the client.

Your application, or client, should maintain a copy of the database by listening to value change notifications from the server device. You can send queries to the server – updating a value in the database, requesting a list of all values from the database after connecting, monitoring for database changes, or requesting the database configuration.

Connection

Establish a connection to the server either by opening a TCP/IP connection, or by connecting to an RS232 COM port on the North device.

The default TCP/IP port number used by DataSync is 1920. Change this to any available port number using the *Network object* within the DataSync driver.

RS232 connections use 19200 baud, no parity, 8 data bits, 1 stop bit, and no flow control. Refer to *Making the Cable* for the details of the RS232 cable.

Message Format

The client sends a query and the server a notification message, as documented in each API method below.

Messages must be formatted as a line of text with ISO-8859-1 character encoding. Each line of text, or message, must finish with the end-of-message marker – represented in this manual using the symbol: ↵. The North device sends a carriage return (control code 0x0D) then a line feed (control code 0x0A) as the end-of-message marker; the application may send either one or both of these characters.

Allow for a maximum message length of 136 characters.

Here is a sample query and notification:

Query

```
LIST↵
```

Notification

```
V0000=12.3↵
V0001=43↵
V0002=1↵
V0003=0↵
V0004=Smart-UPS 450↵
V0005=14/08/14|17:38:38↵
V0006=00:00-08:30,18:00-24:00↵
V0007=00:00=18,09:00=21↵
```

Heartbeats

To monitor for a healthy connection between client and server, a heartbeat message should be sent if either end has not sent another message for 90 seconds.

If either the client or server has not received any message for 3 x 90 seconds, then it should close the connection. The client should then reconnect to the server.

Heartbeat messages are recommended, but optional, and should be enabled within the DataSync driver.

A heartbeat message is an empty message, with only the end-of-message marker sent:

Query

```
↲
```

Notification

```
↲
```

Which Messages to Use

The minimum a client application can implement from the API would be to synchronise values. The application should perform the following tasks:

- Send a *Query: List* on initialisation
- Store values received from *Notification: Value*
- DataSync will continue to send notifications when a value changes

If your client application also wants to synchronise the database structure, then perform the following tasks:

- Send a *Query: Revision* on initialisation, and periodically to check for any database changes
- Check the response in *Notification: Revision*, if the database revision number has changed then send *Query: Type List* to get all objects
- Store objects received from *Notification: Type*, creating a copy of the database
- Once the *Notification: Type* messages have stopped, send a *Query: List* to get all values
- Store values received from *Notification: Value*
- DataSync will continue to send notifications when a value changes
- Check for and send *heartbeats*

Security Considerations

To keep the API simple and stateless, it does not include a method for authentication. Therefore, we recommend DataSync be used over secure private networks, or locally connected devices.

In server mode, the DataSync driver has several features controlling access to it:

- Access can be set to read-only – prohibiting value changes from a client
- Incoming TCP connections can be restricted from a specified IP address only
- Privilege levels limit what data is available from Essential Data/Extra Data, and if it can be adjusted.

Configure all these options using the *Security object* within the DataSync driver.

Database Values

The API presents values from the North device's Essential Data and Extra Data. Essential Data contains 640 values on Commander, and 1280 values on ObSys. Extra Data contains an additional 1024 values. These values are organised in pages, with each page containing 10, 16, 32, or 64 objects, and each object holding a single value.

The values from Essential Data may be accessed using any of the supported API methods. A single address range is used for all functions – 0…639 on Commander, and 0…1279 on ObSys. This object identifier (oid) maps to the standard Essential Data page/object reference as follows:

DataSync oid	Essential Data object reference
0	P1.O1
1	P1.O2
2	P1.O3
3	P1.O4
4	P1.O5
..	..
15	P1.O16

DataSync oid	Essential Data object reference
16	P2.O1
17	P2.O2
18	P2.O3
19	P2.O4
..	..
479	P30.O16
..	..

If Extra Data is used, the extra 1024 values appear in oids that follow on from Essential Data's – i.e. oid 640…1663 on Commander, or 1280…2303 on ObSys.

The maximum objects available from the database can be obtained by reading Database Objects Available (EDC). The number of objects available to the API can be limited by setting Database Size (MV) in *DataSync Setup*.

The oid field is a four-digit number in API methods. For compatibility with DataSync version 1.0, enable Legacy Mode (A.LM) in *DataSync Setup* to restrict this to a three-digit field in the range 0…999.

API Methods

Query: Revision

This method gets the database revision number. The revision number is incremented when objects in the database have been added, removed or edited. The revision number does not change when an object value updates.

A client device should send this query regularly to detect any database record changes. If the revision number has changed, then send the *Query: Type List* message to get a list of all objects.

After the client sends this message, the server will respond by sending a *Notification: Revision* message.

Message Format

Query

```
REVISION↵
```

Example

In this example, the server responds with the revision number of 8.

Query

```
REVISION↵
```

Notification

```
REVN=8↵
```

Query: Type List

This method gets a list of object type descriptions from the database. A client device should send this query when the database has changed. You can monitor for changes in the database by sending the *Query: Revision* message.

After the client sends this message, the server will respond by sending a *Notification: Type* message for each object in its database. The total number of objects in the database is available from Database Objects Available (EDC) in *DataSync Setup*.

Message Format

Query

```
TYPELIST↲
```

Example

In this example, the server responds with all 640 objects in Essential Data.

Query

```
TYPELIST↲
```

Notification

```
T0000=Float;L=UPS status – Load power;U=W;WI=1;D=1↲
T0001=Num;L=UPS status – Battery Time Left;U=mins;WI=1↲
T0002=ENum;L=UPS status – Battery;A=,Healthy,Replace;WI=1↲
T0003=NoYes;L=UPS status - Test;A=No,Yes↲
T0004=Text;L=UPS status – Model;ML=20;WI=1↲
T0005=DateTime;L=UPS status – Date;WI=1↲
T0006=Times;L=UPS status – Alarm;WI=0;P=2↲
T0007=Profile;L=UPS status – Setpoint;WI=0;P=2↲
T0008=↲
..
T0639=↲
```

Query: List

This method gets a list of all object values from the database. A client device should send this query shortly after establishing a connection with a server.

After the client sends this message, the server will respond by sending a *Notification: Value* message for each value in its database.

Message Format

Query

```
LIST↲
```

Example

In this example, the server responds with the eight values available in Essential Data.

Query

```
LIST↲
```

Notification

```
V0000=12.3↲
V0001=43↲
V0002=1↲
V0003=0↲
V0004=Smart-UPS 450↲
V0005=14/08/14|17:38:38↲
V0006=00:00-08:30,18:00-24:00↲
V0007=00:00=18,09:00=21↲
```

Query: Set Value

This method sets a single object value in Essential Data. A client device should send this query when it wants to update the database value held by the server.

After the client sends this message, if the adjustment is successful, then the server will respond by sending a *Notification: Value* message with the updated value. If unsuccessful, no notification message will be sent.

To set a value, the object in Essential Data must be adjustable and have an access security level that permits access. DataSync must also have Enable Value Change object (S.WE) enabled.

Message Format

Query

```
Void=value↲
```

Parameters

Parameter	Value	Description
oid	Number	Four-digit object identifier, in the range 0000…2304
value	Number, String, or List	New value of object. The value depends on the object type configured in Essential Data: Float, Num, ENum – Number OffOn, NoYes – Number: ‘0’ or ‘1’ Text – String (max. 32 chars) DateTime – String in format, ‘dd/mm/yy|*hh*:*mm*:*ss*’ Date – String in format, ‘dd/mm/*yy*’ Times – List of start and end time pairs, ‘*hh*:*mm*-*hh*:*mm*’ Profile – List of time and value pairs, ‘*hh*:*mm*=*value*’

Example

In this example, the query sets oid 3 with the new value ’1’.

Query

```
V0003=1↲
```

Notification

```
V0003=1↲
```

Notification: Value

This method notifies of a single object value change in Essential Data. A server device will send this notification to inform a client of a database value change.

A server will send this notification either unsolicited, or in response to a client sending a *Query: List* message.

Message Format

Notification

```
Void=value↲
```

Parameters

Parameter	Value	Description
oid	Number	Three-digit object identifier, in the range 0000…2304
value	Number, String, or List	New value of object. The value depends on the object type configured in Essential Data: Float, Num, ENum – Number OffOn, NoYes – Number: ‘0’ or ‘1’ Text – String (max. 32 chars) DateTime – String in format, ‘dd/mm/*yy*|*hh*:*mm*:*ss*’ Date – String in format, ‘dd/mm/*yy*’ Times – List of start and end time pairs, ‘*hh*:*mm*-*hh*:*mm*’ Profile – List of time and value pairs, ‘*hh*:*mm*=*value*’

Example

In this example, the unsolicited notification is sent after the value of oid 0 changes to’12.6’.

Notification

```
V0000=12.6↲
```

Notification: Revision

This method provides the Essential Data revision number. The revision number is incremented when objects in the database have been added, removed or edited. The revision number does not change when an object value updates.

A server device will send this notification in response to a *Query: Revision* message.

Message Format

Notification

```
REVN=value↲
```

Parameters

Parameter	Value	Description
value	Number	Database revision number, in the range 1…255

Example

In this example, the server responds with the revision number of 8.

Query

```
REVISION↲
```

Notification

```
REVN=8↲
```

Notification: Type

This method provides an object type description from the database. A server device will send this notification for each object in the database. The total number of objects in the database is available from Database Objects Available (EDC) in *DataSync Setup*.

A server will send this notification in response to a *Query: Type List* message.

Message Format

Notification

```
Toid=type;L=label;U=units;WI=wi;AS=as;ML=maxlen;A=alt;VH=vh;VL=vl;D=dps;P=periods↵
```

Parameters

Parameter	Value	Description
oid	Number	Three-digit object identifier, in the range 0000…2304
type	String	Type of object, one of the following: Text, NoYes, OffOn, Num, Enum, Float, DateTime, Date, Times, or Profile. If the object is unused, then no type or parameters will be present.
label	String	Page and object label, in the format: page label + ‘ - ‘ + object label (max. 43 chars)
units	String	Optional units for object value (max. 8 chars)
wi	Number	Write inhibit. If ‘0’ then can be set using Query: Set Value
as	Number	Optional access security. Two-digit number indicating privilege area and level required
maxlen	Number	Optional maximum value length. Only provided with Text types
alt	String	Optional list of labels for value enumeration. Only provided with Enum types
vh	Number	Optional maximum value. Only provided with Num and Float types
vl	Number	Optional minimum value. Only provided with Num and Float types
dps	Number	Optional number of decimal places to display. Only provided with Float types. Value will always contain this number of decimal places, expanded using ‘0’ if required.
periods	Number	Optional maximum number of on/off times or value/time profiles. Only provided with Times or Profile types

Example

In this example, the server responds with all 640 objects in Essential Data.

Query

```
TYPELIST↵
```

Notification

```
T0000=Float;L=UPS status – Load power;U=W;WI=1;D=1↵
T0001=Num;L=UPS status – Battery Time Left;U=mins;WI=1↵
T0002=ENum;L=UPS status – Battery;A=,Healthy,Replace;WI=1↵
T0003=NoYes;L=UPS status - Test;A=No,Yes↵
T0004=Text;L=UPS status – Model;ML=20;WI=1↵
T0005=DateTime;L=UPS status – Date;WI=1↵
T0006=Times;L=UPS status – Alarm;WI=0;P=2↵
T0007=Profile;L=UPS status – Setpoint;WI=0;P=2↵
T0008=↵
..
T0639=↵
```

Using the Driver

On ObSys and Commander, the DataSync driver is pre-installed. Once started, you will need to set up the driver before you can synchronise with the data.

Making the Cable

Using the RS232 cable specification (Fig. 1), connect the North device COM port to the third-party device's COM port using a null-modem cable. Connector types at each end of the cable are shown.

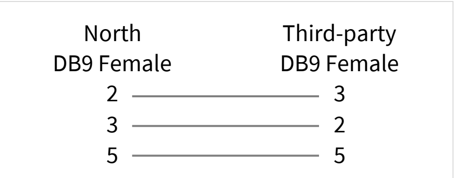

Fig 1 North to Third-party cable

The maximum RS232 cable length is 15m.

Starting the Interface

To start an interface using the DataSync driver, follow these steps:

→ **Start Engineering** your North device using ObSys

→ Navigate to **Configuration, Interfaces,** and set an unused **Interface** to 'DataSync' to start the particular interface

→ Navigate to the top-level of your North device and re-scan it

The driver setup object (Mc), labelled **DataSync Setup**, should now be available.

Setting up the Driver

To set up the driver, follow these steps:

→ Navigate to the **DataSync Setup** object (Mc). For example, if you started interface 1 with the driver earlier, then the object reference will be 'M1'

→ Set **Mode** (OM) to select if the driver should be the server or client device

→ Set **Connection** (CT) to select an RS232 or TCP/IP connection for the API

→ If connection is RS232, set **COM Port** (RS.COM) to select the RS232 port number to use

→ If connection is TCP/IP and mode is client, navigate to the **Network** object (N) and set the **IP address** (IA) of the server.

Checking Communications

After configuring Essential Data, the driver will automatically provide information when requested.

Check the **Connected** object (DS) has a value of 'yes'. In server mode, this indicates a client device has connected. In client mode, this indicates it has connected to the server. Use the **Debug – Last Message** object in Advanced Setup to see the last API message sent by the application.

Object Specifications

Once an interface is started, one or more extra objects become available within the top-level object of the device. As with all North objects, each of these extra objects may contain sub-objects, (and each of these may contain sub-objects, and so on) - the whole object structure being a multi-layer hierarchy. It is possible to navigate around the objects using the ObSys Engineering Software.

Each object is specified below, along with its sub-objects.

Device Top-Level Objects

When an interface is started using the DataSync driver, the objects below become available within the top-level object of the device. For example, if interface 1 is started, then the object reference 'M1' becomes available.

Description	Reference	Type
DataSync Setup Set up the DataSync driver, started on interface *c* (*c* is the interface number)	M*c*	Fixed Container: On the Commander platform this will be *[CDM v20\DataSync v21]* On the ObSys platform this will be *[OSM v20\DataSync v21]*

DataSync Setup

Object Type: [OSM v20\DataSync v21]
Object Type: [CDM v20\DataSync v21]
Object Type: [OSM v20\DataSync v20]
Object Type: [CDM v20\DataSync v20]

The DataSync driver contains the following objects:

Description	Reference	Type
Mode Select operating mode of driver, refer to Detailed Operation for details	OM	Obj\ENum; Adjustable Values: 0=Off, 1=Server, 2=Client
Connection type	CT	Obj\ENum; Adjustable Values: 0=RS232, 1=TCP/IP
Connected Indicates that the driver has a connection open between client and server	DS	Obj\NoYes
Use heartbeats If a message has not been received in 3x90secs, the connection will close. Also sends a heartbeat message if no other message has been sent in 90secs.	HB	Obj\NoYes; Adjustable
Database Objects Available Count of maximum objects available from Essential Data and Extra Data	EDC	Obj\Num
Limit Database size Limits the number of objects available to the API	MV	Obj\Num: 1…2304; Adjustable
COM Port	RS.COM	Obj\Num: 0…8; Adjustable
Restart connection Forces driver to close connection and re-initialize	RST	Obj\NoYes; Adjustable
Network Configure the port number and IP address	N	Fixed Container: On the Commander platform this will be *[CDM v20\DataSync v21\Network]* On the ObSys platform this will be *[OSM v20\DataSync v21\Network]*
Security Control access to the API	S	Fixed Container: On the Commander platform this will be *[CDM v20\DataSync v21\Security]* On the ObSys platform this will be *[OSM v20\DataSync v21\Security]*
Advanced Setup	A	Fixed Container: On the Commander platform this will be *[CDM v20\DataSync v21\Advanced]* On the ObSys platform this will be *[OSM v20\DataSync v21\Advanced]*

Network

Object Type: [OSM v20\DataSync v21\Network]
Object Type: [CDM v20\DataSync v21\Network]
Object Type: [OSM v20\DataSync v20\Network]
Object Type: [CDM v20\DataSync v20\Network]

When connection type is set to TCP/IP, configure the DataSync Network connectivity using this object. By default, all available IP addresses are opened for requests to the API on TCP port 1920.

In server mode, the IP address and TCP Port are used to select which local network interface to use for incoming connections.

In client mode, the IP address is used to specify the location of the remote DataSync server.

Description	Reference	Type
IP address Server mode: Force the driver to use only one of the interface IP addresses. By default, all are used when the address is '0.0.0.0' Client mode: Server IP address to connect	IA	Obj\IP; Adjustable
TCP Port Server mode only. TCP/IP port number. Default 1920	PN	Obj\Num: 1…65535; Adjustable

Security

Object Type: [OSM v20\DataSync v21\Security]
Object Type: [CDM v20\DataSync v21\Security]
Object Type: [OSM v20\DataSync v20\Security]
Object Type: [CDM v20\DataSync v20\Security]

Enable security for the API using this object.

In server mode, objects are available to enable read-only access, restrict incoming TCP/IP connections, and specify privilege levels.

In client mode, an object is available to enable Essential Data changes.

Security Areas and Levels

Within the North security model, there are eight security areas. Security areas could be actual areas in a building, but are normally functional areas – for example, ‘environmental control’ and ‘North engineering’ areas would allow a user to have different privileges in controlling set points and engineering Commanders.

A user is assigned a privilege level in each of the eight areas. The level is in the range zero to seven, seven being the most powerful. When a user wishes to pass a door, his/her privilege level in the door’s area is checked against the minimum required for that area – and then either allowed to pass, or rejected.

The engineer must decide the use of the eight areas. The engineer must also decide the power of the privilege levels. Most systems use only a few levels per area: 0=None, 1=Guest, 2=User, 7=Administrator.

As an example, imagine a page of values in Essential Data. The page needs a user to have a minimum privilege level of 2 in area 1 before it can be viewed. The page is available in a Web browser that checks users with a security database. User A has privilege level 7 in area 1 – she can view the page. User B has privilege level 5 in area 1 – he can also view the page. User C has privilege level 1 in area 1 – she cannot view the page.

The example continues: within this page of values in Essential Data is a temperature set point object. Users need a minimum privilege level of 6 in area 1 to adjust it – therefore User A can adjust the set point, but User B cannot.

Specifying Access Security

Essential Data has Access Security objects to control who can view a page, and who can adjust an adjustable object.

Each Access Security object has a two-digit value. Each controls the access to a particular feature – such as viewing the page, or adjusting the value. The two-digit value is made up of the area digit (1-8), followed by the minimum privilege level (1-7) – for example, if the minimum privilege level is 6 in area 2, then the two digit value is 26. If the value is 00, then no security checks are made.

Privilege Levels in DataSync

The Security object contains a privilege level for each of the eight security areas. The driver uses these to control access to Essential Data and Extra Data when reading or adjusting a value.

Description	Reference	Type
Only accept from IP Server mode only: restrict incoming connections only from this IP address	FIA	Obj\IP; Adjustable

Description	Reference	Type
Enable value change Server mode only: enable write access to Essential Data and Extra Data using the Query: Set Value method	WE	Obj\NoYes; Adjustable
Privilege level in Area *x* Server mode only: the area, x, can be in the range 1…8	P*x*	Obj\Num: 0…7; Adjustable; Default: 7 Range: 0 (no access)…7 (highest privilege level)

DataSync Advanced Setup

Object Type: [OSM v20\DataSync v21\Advanced]
Object Type: [CDM v20\DataSync v21\Advanced]

The DataSync driver contains the following objects:

Description	Reference	Type
Legacy Mode For compatibility with driver version 1.0, sets oid to a three-digit field	LM	Obj\NoYes; Adjustable
Send LIST on Connect In server mode, automatically sends a LIST notification on connecting	SL	Obj\NoYes; Adjustable
Last Message Received The last message received by the DataSync API	DM	Obj\Text
Last Message Sent The last message sent by API. See note 1	SM	Obj\Text
Connection Count Number of TCP connections established via API. See note 1	CC	Obj\Num; Adjustable Write any value to reset
Connection Established Date & time the current TCP connection was established. See note 1	CT	Obj\DateTime
Debug Enable This will store additional debug information in the record file. Use this option only when instructed by North Support	DE	Obj\NoYes; Adjustable

Note 1: object added in build 2022.08.16

Driver Versions

Version	Build Date	Details
1.0	6/8/2014	Driver released
2.0	1/9/2015	Added compatibility with ExtraData and updated Essential Data Added object EDC As total objects is now 2304, notifications now use 4-digit numbers. Added LM (Legacy Mode) to set this to 3-digits. Removed object S.SE
2.1	17/7/2017	Mod: Default connection type set to TCP/IP Mod: Add object A.SL to send LIST notification on server connection Mod: Add object A.DM for last command received Mod: Connected (object DS) reports 'No' on start-up
2.1	23/03/2018	Mod: In Legacy Mode, delay how long an adjusted value is echoed back
2.1	16/08/2022	Mod: Improved handling of TCP SYN attacks from a bad device. Mod: Add Connection Counter (A.CC), Connection Established (A.CT), Last Message Sent (A.SM) objects.

Next Steps…

If you require help, contact support on 01273 694422 or visit *www.northbt.com/support*

North

North Building Technologies Ltd
+44 (0) 1273 694422
support@northbt.com
www.northbt.com

Author: JF
Checked by: TM

Document issued 23/09/2022.