A-Series Telematics Software

Configuration Template Guide

K503225

CONTENTS

About this guide

Aplicom has prepared a set of **template configurations** for **A-Series telematics software**. This guide outlines the templates that are available, with a description and overview of what they do.

Introduction to A-Series Configurations

You can make Aplicom A-Series units function without programming.

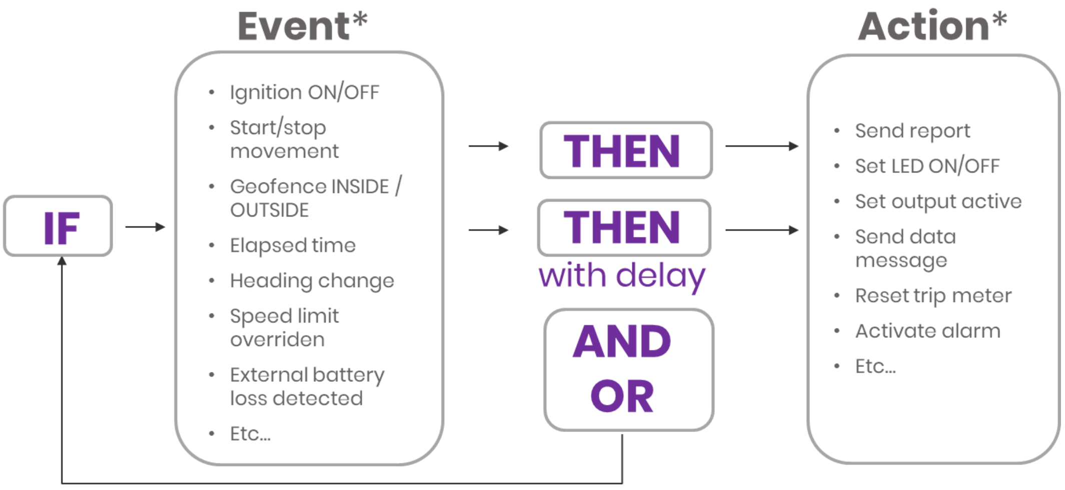
A **configuration** defines the application logic by using the functions of the A-Series **telematics software**. Each configuration is an **XML file** generated with the Aplicom **A-Series SW Configurator** software.

The template configurations can be used as they are, or as a basis for you to create your own application-specific configurations.

You can upload a configuration file from the SW Configurator to an A-Series unit by cable, or remotely over an Aplicom OTAP (Over the Air Provisioning) server. The server installation files and instructions are available to Aplicom partners on the Aplicom Extranet.

The A-Series telematics software follows event-action logic. The configuration defines how and when events are triggered, and what actions are to be taken for each event.

Event-driven logic can be described like this:



* The events and actions shown here are examples only. For a full list, please refer to the Aplicom A-Series SW Configurator User's Manual

Without a configuration, the Aplicom A-Series telematics software doesn't do anything. A configuration is always needed on top of it.

A-Series units are delivered with a basic tracking configuration as default, unless you order another configuration for factory installation.

The reason for configuration templates

Templates offer you a quick start to creating functionality for your own use cases on the A-Series devices. Use the template as the basis for your own applications by modifying it and adding new functions.

A configuration template can be used without any changes for testing or demonstrating A-Series functionality in practice. You can also use templates as a learning tool, to understand what configurations look like in real life.

Unleash the full power of Aplicom devices

We would like to help you take full advantage of our versatile A-Series products. We will train you, describe what the A-Series can do, and show you the best ways to create configurations that work well.

Tailored specially for your needs, our training workshops give you the opportunity to work on real-life configurations with hands-on support from our experts. You will learn to use configurations quickly, flexibly and efficiently in response to the requirements of your customers.

Pre-installation of configurations

We can upload configurations to A-Series units before you have them installed. You can provide the configuration file yourself, or you can ask us to create it for you, according to your specifications. Please contact our sales team for details.

Do I always need to create my own configurations myself?

No, you don’t. Tell us what you want, and we can do it for you!

We have created a lot of configurations for many different applications. Because we have done so many, we are highly skilled at this and can do it quickly and cost-effectively. Please contact our sales team for a quote!

Technical notes

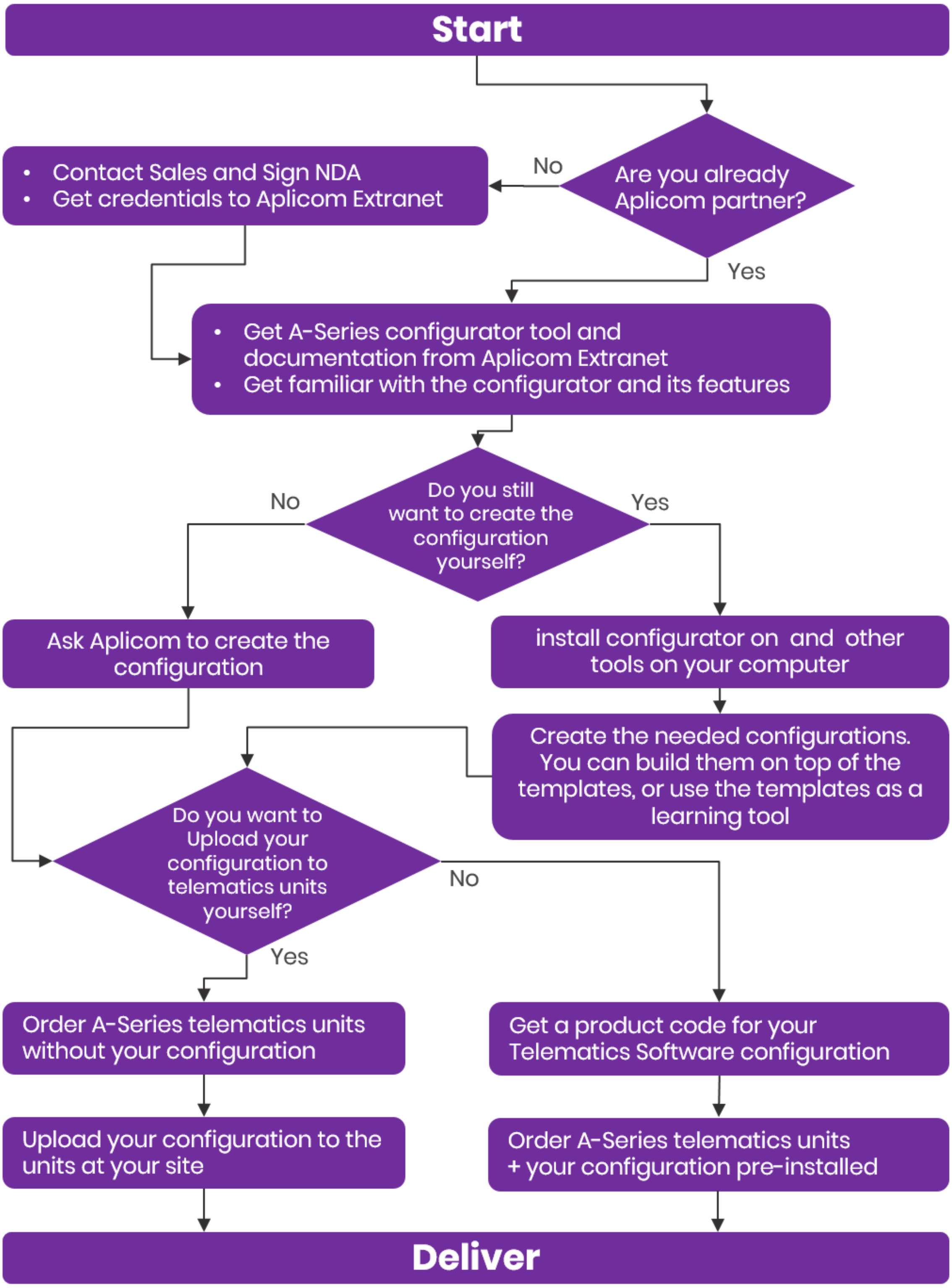
Many templates provide similar functionality on both the A9 and A11 units. However, because they use different software interfaces, their configuration files are different, even when their functions are the same.

The full functionality of the A-Series Configurator Tool is described in the **Aplicom A-Series SW Configurator User’s Manual**.

To aid clarity, the present document shows the names of events and actions in `a different font` without underscore marks, so they look like this: `EBS data event.` In the actual XML file and the User’s Manual, they are written in upper case and with underscores, like this: `EBS_DATA_EVENT`.

Event handlers execute your code by using events. You can use the ones that are included in the templates, or you can create your own ones.

Process for getting started



Start

Are you already Aplicom partner?

No:
- Contact Sales and Sign NDA
- Get credentials to Aplicom Extranet

Yes:
- Get A-Series configurator tool and documentation from Aplicom Extranet
- Get familiar with the configurator and its features

Do you still want to create the configuration yourself?

No: Ask Aplicom to create the configuration

Yes: install configurator on and other tools on your computer

Create the needed configurations. You can build them on top of the templates, or use the templates as a learning tool

Do you want to Upload your configuration to telematics units yourself?

Yes: Order A-Series telematics units without your configuration

Upload your configuration to the units at your site

No: Get a product code for your Telematics Software configuration

Order A-Series telematics units + your configuration pre-installed

Deliver

A-Series Default Tracking Template

Configuration files

AplicomA11-ExampleConfig-2.0.xml for A11
AplicomA9-TRIX-ExampleConfig-2.0.xml for A9

Purpose

Basic tracking

These configurations are the default delivery configurations for telematics software of Aplicom A9 and A11 devices.

How it works

This configuration has two parameter sets: “default” for low speed range and “hiway” for high speed range. The Speed limit over/below event is used to select the desired parameter set to be applied.

The parameters triggering the Direction changed, Start moving and Stop moving events have different values in these two sets.

Connection-related parameters (default Aplicom SIM) and the server address can be set with an SMS message, after which the device is ready for action.

Example SMS for sending parameters:

A11+CONFuser=*aplicom*,pass=*m2m572,*dns=*0.0.0.0*,apn=*apn.aplicom.com*,tcpAddr=*ads.aplicom.com:4443*

Note! Replace the variable values written on italic above to ones suitable for your operator and server.

Ignition ON turns the unit ON.
The configuration sends D-protocol snapshots with Send snapshot action on the following events:
- Software start
- Ignition on and Ignition off
- Scheduled event every 180s
- Direction changed: event triggered only if the moving flag is set in the Start moving event
- Distance travelled
- Start moving and Stop moving
- Speed limit over and Speed limit below: the applied parameter also changes
- GPS fix status changed: GPS fix status acquired / lost
- AD threshold: main power supply below the default threshold
- Battery low: warning level
- iButton: key logging in
- iButton: key logging out
- System info: version information collected

The configuration uses A11 device LEDs for the following notifications:

- LED A: off when no network connection, red when in roaming net, green when in home net
- LED B: red when no GPS fix, green when valid GPS fix
- LEDs E and F are hardcoded: please see the A11 Installation Guide for details
- In A9 device the LED is red when no GPS fix, green when valid GPS fix.

Tracking + FMS Template 1 for A9 and A11

Configuration files

AplicomA11-ExampleConfigFMS-2.0.xml
- Tracking and FMS functions for A11

AplicomA9-TRIX-ExampleConfigFMS
- Tracking and FMS and functions for A9

Purpose

Adding of FMS to CAN1 interface to the functions of **AplicomA11-ExampleConfig-2.0.xml** and **AplicomA9-TRIX-ExampleConfig-2.0.xml**

Functions

In addition to AplicomA11-ExampleConfig, it connects to the FMS CAN interface, polls the data in 10s intervals, and sends F-protocol snapshots of FMS events and time based on scheduled events.

How it works

An F-protocol type FTCP transport is added to the transports section. The server address is configured as a variable that can be set with an SMS message.

`Send CAN FMS snapshot` action is used to send the F-protocol snapshot

- in `Scheduled event` every 180s
- in `FMS-CAN overspeeding event`
- in `FMS-CAN harsh braking event`
- in `FMS-CAN engine overtemperature event`
- in `FMS-CAN engine overrevolutions event`
- in `FMS-CAN cruise control event`

The parameters for the FMS-CAN interface and triggering levels for the events are defined in the CAN interface 1 section of the configuration.

Trailer Tracking Template with EBS for A9 IPEX

Configuration files

AplicomA9-TRIX-ExampleConfigEBS2.0.xml
- Tracking and EBS functions for A9 IPEX

Purpose

Adds EBS to the CAN interface to the functions of **AplicomA9-ExampleConfig-2.0.xml**

Functions

In addition to the functions of AplicomA9-ExampleConfig-2.0.xml, this configuration connects the device to the EBS interface of a trailer. It sends EBS data in D-protocol snapshots to a server. The data is located in the “event specific bytes” section of the snapshot.

How the EBS functions work

“EBS - trigger timer” alarm polls the EBS data every 60s. This alarm is activated in “EBS - Software start” event. In this configuration, the `Alarm active` event handler of this alarm is named “EBS – trigger timer activated”.

In the `Alarm active` event handler, “EBS – trigger timer activated”, the EBS data is polled by using `Trigger EBS data event` action.

That action will trigger an `EBS data event`. The name of the event handler is “EBS –EBS data event for timed trigger”. The available polled EBS data is then sent in D-protocol snapshots event-specific bytes section by using `Send snapshot` action.

The data blocks included in the sent EBS data are defined in the CAN interface configuration: Monitored EBS data blocks.

A change in Monitored EBS events defined in the CAN interface configuration trigger is also an `EBS data event.` The event handler of that event has individual source parameters listed, but not the ‘Triggered by a Trigger EBS data event action’ selection box selected. The name of the event handler is EBS –EBS data event for detected events. This event handler also uses the `Send snapshot` action for sending a D-protocol snapshot with the EBS data to the server.

Advanced Template 1 for A11, with FMS, EBS, K-Line and real-time Tacho interface

Configuration files

AplicomA11-ExampleConfigFMS-EBS-Tacho-2.0.xml
- Tracking and FMS, EBS, K-Line and real time tacho functions for A11

Purpose

Adds K-line tachograph interface and ”real-time” tachograph data to the functions of AplicomA11-ExampleConfigFMS-EBS-2.0.xml

Functions

In addition to the AplicomA11-ExampleConfigFMS-EBS-2.0.xml configuration, this configuration uses the K-line interface to connect to a Siemens VDO DTCO tachograph and sends ”real-time” tachograph data in E-protocol snapshots to a server.

How it works

The E-protocol type ETCP transport is added to the transports section. The server address is configured as a variable which can be set with an SMS message.

The `Send tachograph snapshot` action is used to send the E-protocol snapshot whenever `Tachograph event` is triggered.

The tachograph events are:
- *Card 1 insertion status changed*
- *Card 2 insertion status changed*
- *Driver 1 working state changed*
- *Driver 2 working state changed*
- *Driver 1 time related warning*
- *Driver 2 time related warning*
- *Over speeding status changed*
- *Ignition status changed*
- *Vehicle motion status changed*

The connected tachograph is selected in the Tachograph section of the configuration.

Advanced Template 2 with FMS and RDL

Configuration files

AplicomA11-ExampleConfigFMS-RDL2.0.xml
- Tracking, FMS and RDL functions for A11
AplicomA9-TRIX-ExampleConfigRDL-2.0.xml.
- Tracking and RDL functionality for A9 TRIX

Purpose

Adding of RDL functionality to the functions of **AplicomA11-ExampleConfigFMS-2.0.xml** and **AplicomA9-TRIX-ExampleConfigRDL-2.0.xml**

RDL functions

This configuration periodically opens a connection to the Aplicom RDL server and enables a remote download of tachograph data.

It also detects if a driver card hasn't been read within a set time. If not, it will request a download of the card data from the server.

How RDL functions work

"RdlServer" transport is added to the transports section. This transport is used for sending messages containing the device IMEI to the server, thus opening the TCP/IP connection so it can send commands to the device.

"RdlAlive5minTimer", "RdlFourHourTimer" and "RDLaliveTimer" Alarms are used to trigger sending those messages to the server so that the connection is opened for 5 minutes after the `Ignition on` event and once every four hours during continuous use.

The RDL server initiates the RDL by sending the START_TACHO_RDL command to the device. The Data Event "RdlDownloadTachoRequest" receives the command and launches the download attempt. The RDL server sends the command according to user-defined conditions in the RDL server or a request from the device.

The device will send a request to the RDL server if a `Tachograph download status changed` event is triggered with download status NEW_DRIVER_CARD, when a new or unread card is inserted into the driver card slot in the tachograph. It will also send a request with download status NEW_CO_DRIVER_CARD when a new or unread card is inserted into the co-driver card slot in the tachograph. The card is considered unread if it has not been read within the last seven days. The time that defines "unread" is configurable.

The command to start the operation always comes from the RDL server.

The operation consists of three main steps:

1. Authentication
2. Reading files from the tachograph to the A11 or A9
3. Sending files to the RDL server

Steps 1 and 2 do not require any configuration files in the A11 or A9. The `Tachograph download status changed` event called “RdlTachoDownloadStatusChangeSend” sends snapshots with progress information to a transport defined by a variable “%transportInfo”. For example, DTCP transport might be selected. The download status is shown in the TACHO_DL snapshots Event information field as a signed integer (for more information, see the A11 SW User’s Manual, Chapter 10.48 TACHO_DL_EVENT).

The RDL server is notified when the requested files are successfully read from the tachograph and saved to the A11 or A9. The server then uploads the files by sending a START_TACHO_UPLOAD command to the device. The `Data event` “RdlStartFileUpload” receives the command and uses an `Upload file to server(HTTP)` action to send the files to the server. The RDL server can request the directory listing prior to the upload to locate the files in the A11 or A9 flash file system.

If the new driver card insertion detection option is to be used, then the tachograph device type should be specified, and the tachograph should be connected to A11 or A9 by a K-line or RS485 connection.

For testing purposes (e.g. after an installation), it is possible to trigger the RDL request to the server from the A11 or A9 by sending a TEST_TACHO_RDL SMS to it.

When using the Aplicom RDL service and server with any adaptation of RDL functionality to a new configuration, it is strongly recommended to copy those configuration items starting with RDL, Rdl or rdl, just as they are in this example.

Advanced Template 3 with FMS and BLE (Bluetooth Low energy)

Configuration files

AplicomA11-ExampleConfigFMS-BLE-2.1.xml
- Tracking, FMS and BLE functions for A11

Purpose

Adding of BLE functionality to the functions of **AplicomA11-ExampleConfigFMS-2.0.xml**

The example configuration presents example functionality to scan BLE beacons more frequently when the vehicle is stopped for loading and unloading, when the changes in the heard beacons are typically changing due to adding or unloading assets on board. At these moments data is sent frequently to see changes as soon and as accurately as possible. Instead, during long drivers with higher speed the scanning of BLE beacons is done with longer time periods. The situation with heard beacons stabilizes to beacons on board.
This method saves SIM communications costs efficiently compared to configuration that sends continuously at high scanning rate.

BLE functions

This configuration listens available Bluetooth beacons and periodically sends the list of available beacons and their broadcasted data in G-protocol snapshots Group 12 – Event, event specific bytes.

How BLE functions work

Beacon scan is enabled in configurations BLE section and all heard beacons with applicable signal strength of -90dB are accepted. The idea is to read the beacon data and trigger the *BLE beacon found* event with *Trigger BLE beacon read* action on different time intervals depending on the motion state of the vehicle and thus the *Scanned beacons read interval* is set to 86400s (24h).

The list of heard beacons is reset at *Software start.*

“BeaconQuickScanTimer” *alarm* is activated in *Ignition on* and it is triggering “BeaconQuickScanTime” *Alarm active* type events first one event in 15s then on 30s interval. The event handler uses “ScanNearbyBeacons” action to read the heard beacons and to trigger “BeaconData” event which then sends a G-protocol snapshot containing the beacon data to server.

When the vehicle starts moving the “BeaconQuickScanTimer“ is stopped and the “BeaconSlowScanTimer” is activated in “StartMoving” event. “BeaconSlowScanTimer” triggers the “BeaconSlowScanTime” *Alarm* once in 30s and then on 900s interval. The event handler uses “ScanNearbyBeacons” action to read the heard beacons and to trigger “BeaconData” event which then sends a G-protocol snapshot containing the beacon data to server.

When the vehicle stops the “BeaconSlowScanTimer” *alarm* is stopped and “BeaconQuickScanTime” *alarm* is activated and thus the heard beacons are sent to server on shorter time interval again.

Both alarms are deactivated in *Ignition off* event.

How to do it - a practical example with screen shots

Here is a simple practical example of a workflow to create a configuration.

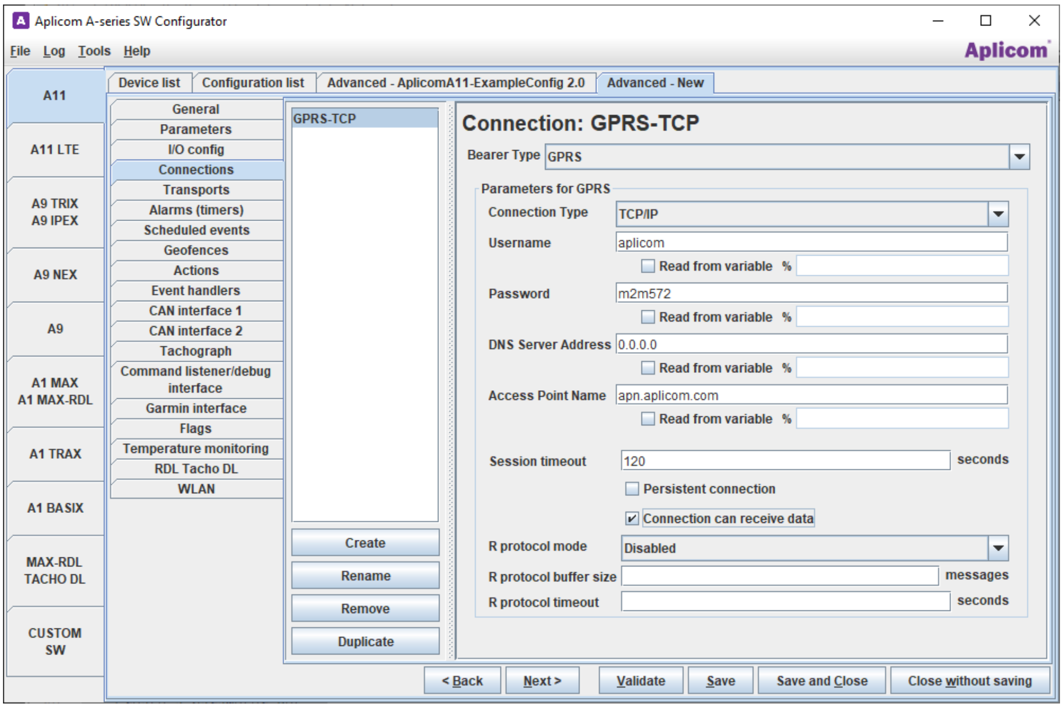
This example describes a simple configuration that sends a **snapshot** to a server with a defined interval of travelled distance.

Please note, this is an example only, not instructions. For detailed information, please refer to user documentation:

- K530100 A11 telematics sw user manual
- K530800 A series sw configurator user manual

1. Connections section

Select Bearer type		
Select Connection type		
Define operator related parameters	Username	
	Password	
	DNS	
	APN	
Select connection persistency	Select connection persistency	Connection opened and closed for each TCP/IP frame separately
	Session time out	Define the time the device is keeping session open if nothing is sent
	Persistent connection	The device is keeping session open
Select if R-protocol is used	Select, Disabled or ACK or NACK mode	
	If ACK or NACK mode selected	Define the parameter values



Connection: GPRS-TCP

Bearer Type: GPRS

Parameters for GPRS

Connection Type: TCP/IP

Username: aplicom

□ Read from variable %

Password: m2m572

□ Read from variable %

DNS Server Address: 0.0.0.0

□ Read from variable %

Access Point Name: apn.aplicom.com

□ Read from variable %

Session timeout: 120 seconds

□ Persistent connection

☑ Connection can receive data

R protocol mode: Disabled

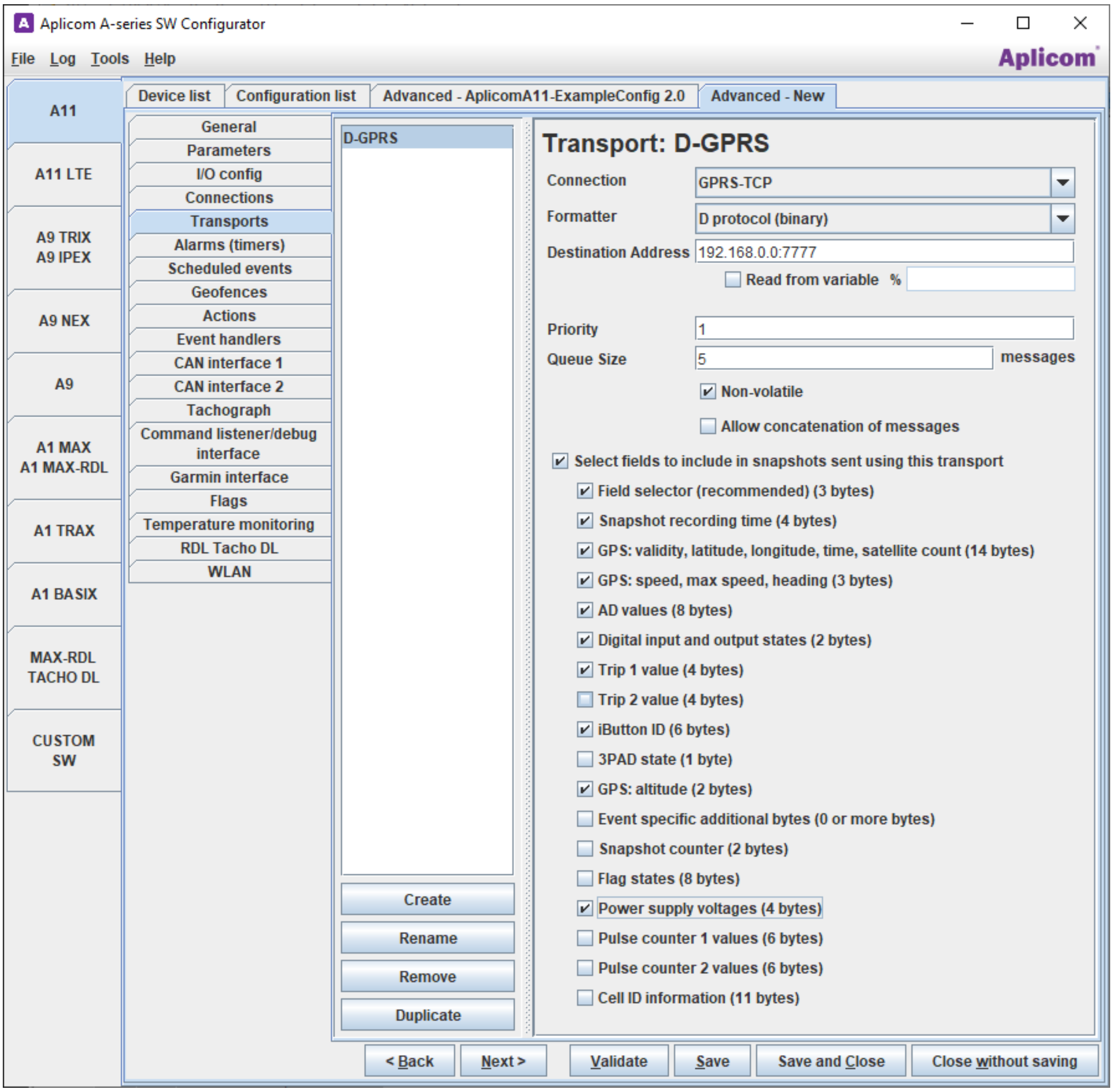
R protocol buffer size: messages

R protocol timeout: seconds

2. Transports section

Create a Transport.

Select	The connection that this transport uses	
Select	Formatter i.e. the snapshot protocol	
Define	Destination address where the snapshots are sent to.	
Define	Priority level	
Define	Snapshot sending queue length.	
Select	Volatile / non-volatile transport.	
Select	Concatenation of snapshots allowed / not allowed	
Select	Fields included into the sent snapshots	



Transport: D-GPRS

Connection: GPRS-TCP

Formatter: D protocol (binary)

Destination Address: 192.168.0.0:7777

□ Read from variable %

Priority: 1

Queue Size: 5 messages

☑ Non-volatile

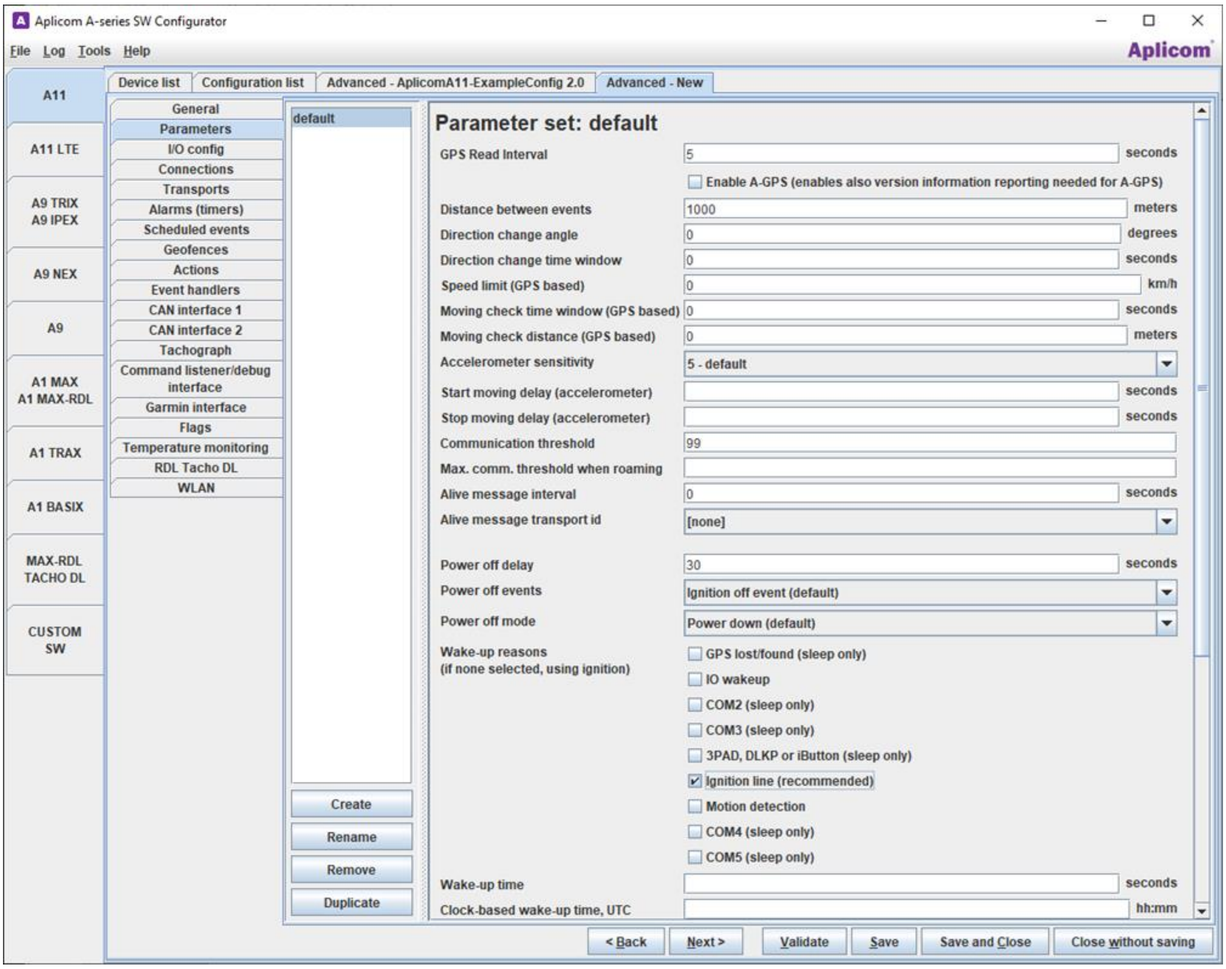
□ Allow concatenation of messages

☑ Select fields to include in snapshots sent using this transport

- ☑ Field selector (recommended) (3 bytes)
- ☑ Snapshot recording time (4 bytes)
- ☑ GPS: validity, latitude, longitude, time, satellite count (14 bytes)
- ☑ GPS: speed, max speed, heading (3 bytes)
- ☑ AD values (8 bytes)
- ☑ Digital input and output states (2 bytes)
- ☑ Trip 1 value (4 bytes)
- □ Trip 2 value (4 bytes)
- ☑ iButton ID (6 bytes)
- □ 3PAD state (1 byte)
- ☑ GPS: altitude (2 bytes)
- □ Event specific additional bytes (0 or more bytes)
- □ Snapshot counter (2 bytes)
- □ Flag states (8 bytes)
- ☑ Power supply voltages (4 bytes)
- □ Pulse counter 1 values (6 bytes)
- □ Pulse counter 2 values (6 bytes)
- □ Cell ID information (11 bytes)

3. Create a Parameter set

Define	GPS read interval.	
Define	Distance between events.	
Define	Power off delay	
Select	Power off events.	
Select	Power off mode	
Select	Wake up reasons	



Parameter set: default

GPS Read Interval: 5 seconds

□ Enable A-GPS (enables also version information reporting needed for A-GPS)

Distance between events: 1000 meters

Direction change angle: 0 degrees

Direction change time window: 0 seconds

Speed limit (GPS based): 0 km/h

Moving check time window (GPS based): 0 seconds

Moving check distance (GPS based): 0 meters

Accelerometer sensitivity: 5 - default

Start moving delay (accelerometer): seconds

Stop moving delay (accelerometer): seconds

Communication threshold: 99

Max. comm. threshold when roaming:

Alive message interval: 0 seconds

Alive message transport id: [none]

Power off delay: 30 seconds

Power off events: Ignition off event (default)

Power off mode: Power down (default)

Wake-up reasons (if none selected, using ignition):

- □ GPS lost/found (sleep only)
- □ IO wakeup
- □ COM2 (sleep only)
- □ COM3 (sleep only)
- □ 3PAD, DLKP or iButton (sleep only)
- ☑ Ignition line (recommended)
- □ Motion detection
- □ COM4 (sleep only)
- □ COM5 (sleep only)

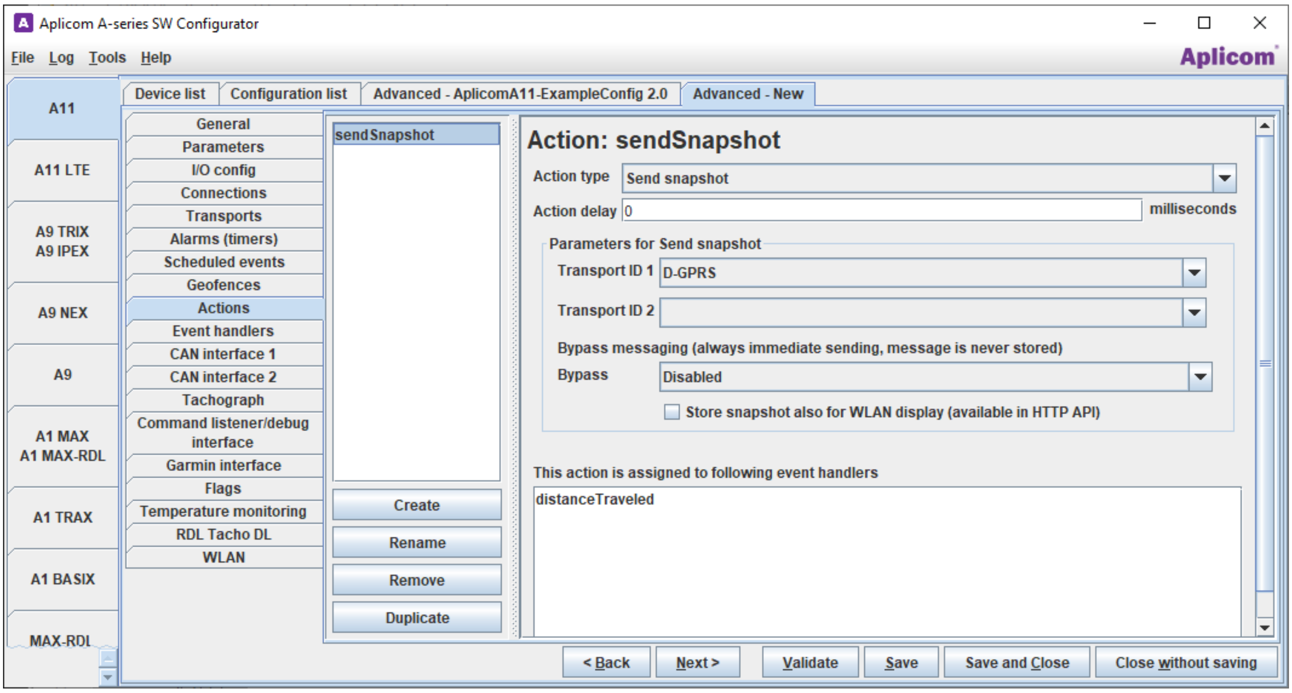
Wake-up time: seconds

Clock-based wake-up time, UTC: hh:mm

4. Actions

Create a “Send snapshot” type action

Select	Transport ID 1	Here you give a name to the action that you create. In this example it is “sendSnapshot”



Action: sendSnapshot

Action type: Send snapshot

Action delay: 0 milliseconds

Parameters for Send snapshot

Transport ID 1: D-GPRS

Transport ID 2:

Bypass messaging (always immediate sending, message is never stored)

Bypass: Disabled

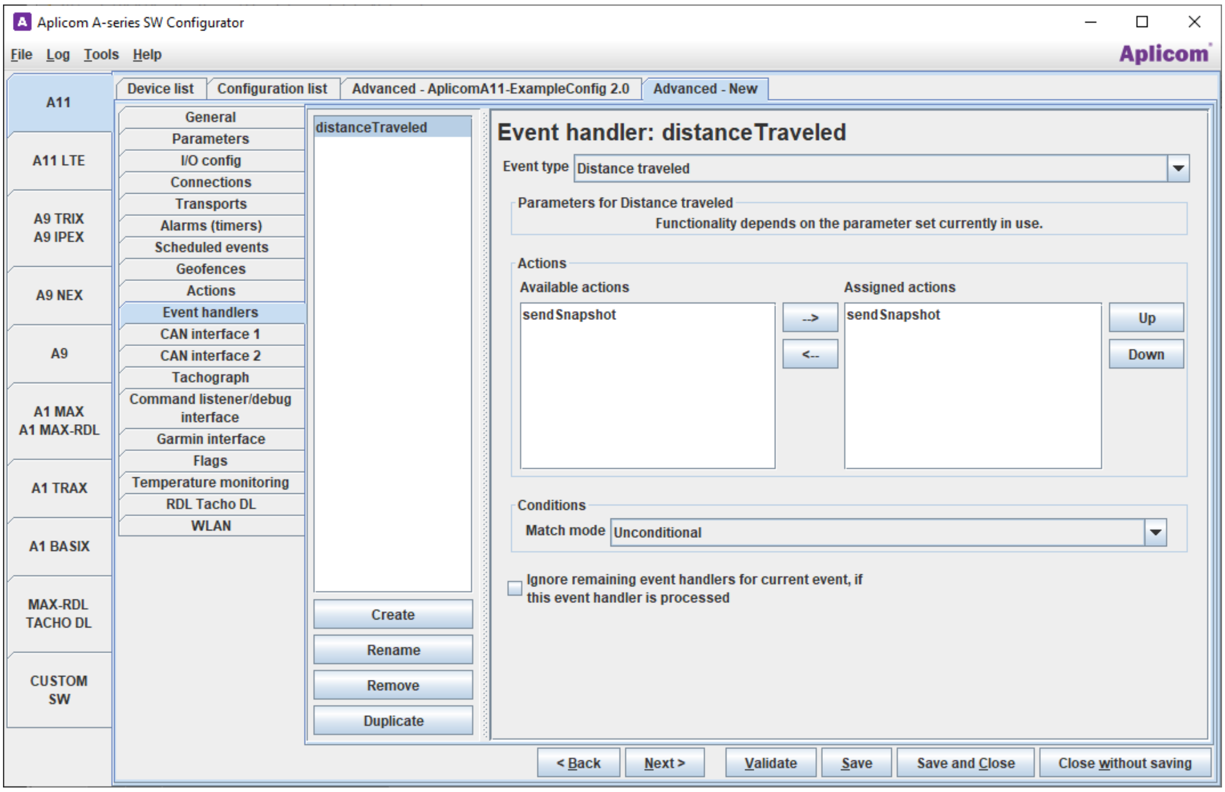
☐ Store snapshot also for WLAN display (available in HTTP API)

This action is assigned to following event handlers

distanceTraveled

5. Event handlers

Create a ”Distance traveled” type event handler	
Select and assign the **sendSnapshot** action	You have defined this name in the previous step



Aplicom A-series SW Configurator

Event handler: distanceTraveled

Event type: Distance traveled

Parameters for Distance traveled

Functionality depends on the parameter set currently in use.

Actions

Available actions: sendSnapshot

Assigned actions: sendSnapshot

Conditions

Match mode: Unconditional

☐ Ignore remaining event handlers for current event, if this event handler is processed