DashBoard Application Note:
Methods for OGP Developers to Speed Up Device Initialization

This application note is for OGP-Binary developers who may wish to use one of the three methods described below to handle the initialization of devices with many parameters quickly and efficiently. This involves bypassing the typical DashBoard response message to each device query that is described in the DashBoard openGear Developer Guide's device communication model. The three methods are listed here to help you choose which method will work best for your needs:

- **OGP Blast Mode** —This method is recommended when a device is running its own OGP TCP/IP server, or is in an openGear Frame with relatively low CAN Bus utilization.
- **XML Side Loading of the Device Description** — This method requires dynamic regeneration of an XML document from the OGP device and requires the generation of a potentially large XML document.
- **Instantiate a New Connection to the Device** — This method involves an extra TCP/IP connection, and the loss of DataSafe and SNMP usage capabilities provided by the openGear Frame.
  Note: This method allows the dedicated connection to use either OGP-Binary or OGP-JSON. If you choose to continue using OGP-Binary, you can pair this with OGP blast mode.

Note: For more information on the methods provided in this guide, read the ***openGear Development Guide Part II – Software (8200DR-006-9.5)***.

OGP Blast Mode "init"

The OGP Blast Mode allows a device to indicate to DashBoard that it is about to initiate a "blast" of messages, instead of a typical device request/response message flow. In this case, DashBoard will allow the OGP-Binary device to send a number of response messages without waiting for their requests.

To use OGP Blast Mode, it is required that the OGP device pre-emptively sends its responses to DashBoard's typical request query in advance, since the DashBoard client still needs this information.

You should be able to configure blast mode by turning it on or off for your device. It must default to ‘off’.

OGP_GET_NUMPARAMS Response

Message type: 0xC5
Message length: 3*
Response required: none

Description: This is the required response to an ***OGP_GET_NUMPARAMS*** request. It provides the number of parameters supported by the device.

Field	offset	length	format	Description
rc	0	1	uint8	return code OGP_OK – normal return
numPar	1	2	uint16	number of parameters for this device
xmlDescriptionLen	3	1	uint8	length of xmlDescriptionURL (including null-terminator) to follow
xmlDescriptionURL	4	*	string	URL of a device definition XML file (if a URL is provided, the remainder of the parameter descriptor and menu queries are skipped by DashBoard)
numStringOIDs	4 + xmlDescriptionURL Length	4	int32	number of parameters with String-based OIDs for this device
blastSupportFlag	8 + xmlDescriptionURL Length	1	uint8	Flag to indicate whether this device supports OGP_BLAST_REQUEST message

OGP_START_INIT_BLAST Request

Message type: 0x4E
Message length: 2
Response required: ***OGP_START_INIT_BLAST Response***

Description: This informs the device that it is free to broadcast its responses to the standard DashBoard initialization queries immediately. This allows the device to avoid round-trip delays by sending its parameter descriptors, parameter values, and menus before DashBoard requests them.

Field	offset	length	format	Description
spare	0	1	uint8	placeholder for return code
state	1	1	uinit8	OGP_START (0x00)

OGP_START_INIT_BLAST Response

Message type: 0xCE
Message length: 2
Response required: none

Description: This is the required response to OGP_START_INIT_BLAST it indicates that a blast will start or that a blast has been denied.

Field	offset	length	format	Description
rc	0	1	uint8	return code **OGP_OK** – normal return (OGP Blast will follow) **OGP_UNSUPPORTED (0x01)** – OGP Blast disabled or not supported **OGP_RESPONSE_DENIED (0x03)** – OGP Blast already in progress
state	1	1	uinit8	OGP_ START (0x00) Or OGP_COMPLETE (0x01)

To implement the OGP_START_INI_BLAST RESPONSE

1. An initialization blast consists of the following messages:

```
TX: OGP_START_INIT_BLAST Response with OGP_START (0xCE)
TX: OGP_GET_DESCRIPTOR_Response (0xC7) for first OID
TX: OGP_GET_PARAM Response (0xC9) for first OID
```

2. Make sure to repeat for each OID in the param list in order, as shown below.

Note: You can also (optionally) pre-send your responses to the menu requests.

```
TX: OGP_GET_MENUSET_NAME Response (0xD0) for menu set 0
TX: OGP_GET_MENU_COUNT Response (0xD1) for menu set 0

TX: OGP_GET_MENU_NAME Response (0xD2) for first menu
TX: OGP_GET_MENU_OIDS Response (0xD3) for first menu
TX: OGP_GET_MENU_STATE Response (0xDA) for first menu
```

3. Repeat for each menu in the menu set, as shown below.

Tip: This includes menu sets 1 and 2.

```
TX: OGP_START_INIT_BLAST Response with OGP_COMPLETE (0xCE)
```

IMPORTANT:

A device responding to an initialization blast through an openGear Frame must limit its response rate to avoid interfering with normal communications on the CAN bus. A response rate of 80 messages per second (or less) is required.

If additional OGP_START_INIT_BLAST messages are received during the blast, an OGP_START_INIT_BLAST Response must be sent with OGP_RESPONSE_DENIED

If the device receives a request message, it must respond to it even it has already sent an appropriate response as part of its blast.

XML Side Loading of the Device Description

This method of speeding up device initialization involves adding an extra bit of information to the device response for the number of OIDs (*OGP_GET_NUMPARAMS* Response). It involves providing a URL for an XML device description. If it is provided, then DashBoard stops asking for the device description, and instead retrieves it from the specified XML document. It must contain all the device descriptor information including menu groups, menus, parameters, parameter values, and constraints, along with any other information typically provided in the device description.

You will need to dynamically generate the XML document from the OGP device to ensure the parameter values are read by the device. Normal parameter query messages are skipped, and DashBoard will not request the parameter value from the device. Instead, DashBoard will use the "value" attribute of the XML parameter tags as the parameter values until additional updates to the value are sent asynchronously via OGP-Binary by the device.

Troubleshooting Tip: If the URL is defined, DashBoard will attempt to fetch the XML document and use it to initialize the device. If the ethernet settings are known to be incorrect, the device should not provide this URL. If the XML cannot be reached, the device will fail to initialize using the XML side load method. As of DashBoard 9.5, if the XML file can not be accessed, DashBoard will default to using the OGP-Binary querying method to initialize the device. You can choose to continue device initialization using the OGP-Binary querying method, or update your ethernet settings to resume using the XML side loading initialization method.

To create an XML Document from the OGP Device

- In the DashBoard tree view, simply right-click on the OGP device and select **Save Configuration to File**. An example of what the XML device definition would look like for your device will be saved as an openGear Device (“OGD”) file in the location of your choice.
- You can use this XML file to prevent DashBoard from sending requests for the device description, but the XML file must contain all the definitions typically provided in the device description.

Below is an outline of the XML file structure:

```
<?xml version="1.1" encoding="UTF-8"?>
      <frame>
      <card>
            <params>
                  <param/>
                  <param/>
```

```
            . . .
        </params>
        <statusmenu>
            <menu>
                <param/>
                <param/>
                . . .
            </menu>
            <menu/>
            . . .
        </statusmenu>
        <configmenu>
            <menu>
                <param/>
                <param/>
                . . .
            </menu>
            <menu/>
            . . .
        </configmenu>
        <menugroup>
            <menu>
                <param/>
                <param/>
                . . .
            </menu>
            <menu/>
            . . .
        </menugroup>
        <menugroup/>
        . . .
    </card>
    <card/>
    . . .
</frame>
```

Instantiate a New Connection to the Device

This method involves instantiating a new connection to the device directly using the Device Index URL reserved OID and creating an additional node to create and access this connection under your device's original node in the DashBoard Tree View.

The XML document specified by the Device Index URL reserved OID allows additional properties or entirely new nodes/levels of hierarchy to be specified under a device node in DashBoard's tree view. If one of the new nodes under the device is a connection to an OGP server, DashBoard will connect to this new server and place a "frame" node under the device node.

Note: The device connection is separate and is applicable if you are using an OGP binary device that lives inside of the OpenGear frame talking over the CAN bus. If you already have a dedicated connection, use either **OGP Blast Mode** or **XML Side Loading of the Device Description** to speed up device initialization, or switch into OGP-JSON.

To create a new device node using the Device Index URL

1. Create the XML file on the OGP device containing the information in the string below. See below for a table of required fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties version="1.0">
     <children>
          <child>
               <entry key="equipmentType">editor-link</entry>
               <entry key="child-id">basic-menu</entry>
               <entry key="node-name">Default Menu System</entry>
          </child>
          <child>
                              <comment>DashBoard Device Connection Settings</comment>
                    <entry key="serviceUrl">service:broadcast-equipment</entry>
                    <entry key="equipmentType">opengear</entry>
                    <entry key="address">172.16.9.31</entry>
                    <entry key="port">5253</entry>
                    <entry key="node-id">device</entry>
                    <entry key="node-name">Jim's Frame</entry>
             </child>
     </children>
</properties>
```

Field	Description
equipmentType	You may use equipmentType **opengear** to connect to the device using binary OGP messaging, or equipmentType **opengear-json** to connect to the device using OGP-JSON messaging.
address	Your IP address.
port	The port that your OGP server is running on.
node-id	A globally unique identifier for your device. It is recommended that you use a manufacturer serial number.
node-name	The node will appear with this name.

2. Populate the value of the Device Index URL into the reserved OID 0xFF0D in DashBoard.

This will create a device node under the CAN bus which contains two nodes within it.
Editor Link: This node points to the interface for the device node and provides access to the device parameters served over the CAN bus.
Device Connection Settings: This node contains the connection information which establishes a dedicated connection to the device in a new **frame node**.

Additional navigation improvement

Once the new **frame node** is created, there are two additional reserved OIDs that can be specified for ease of navigation. By specifying these OIDs, you can turn the **frame node** into a clickable **leaf node**, which will allow you to navigate directly to the operator interface by double-clicking on the frame.

To turn the frame node into a clickable leaf node

1. Navigate to the reserved OID CONFIG_SLOT with the value FE07 and enter the value of the slot to be opened when the frame is opened.
2. Navigate to the reserved OID DISPLAY_OPTIONS with the value FF0B.
3. Under DISPLAY_OPTIONS_INDEX_HIDDEN, enter the same slot value.

Once you have completed these steps, double clicking on the frame will allow you to navigate directly to the operator interface.