

Variable	Type	Description
ulDest	UINT32	0x20
ulLen	UINT32	0
ulSta	UINT32	See section Status and Error Codes
ulCmd	UINT32	0x2F87

Table 24. Confirmation packet *HIL_SET_FW_PARAMETER_CNF_T*

Example

The following example shows how to set a value as identification value:

```
void FillOutFwParamDeviceIdentPacket(TLR_UINT32 ulSrc, HIL_SET_FW_PARAMETER_REQ_T* ptPkt, TLR_UINT16 usIdentValue)
{
    ptPkt->tHead.ulCmd = HIL_SET_FW_PARAMETER_REQ;
    ptPkt->tHead.ulExt = 0;
    ptPkt->tHead.ulSta = 0;
    ptPkt->tHead.ulSrcId = 0;
    ptPkt->tHead.ulSrc = ulSrc;
    ptPkt->tHead.ulLen = 12;
    ptPkt->tHead.ulRout = 0;
    ptPkt->tHead.ulId = 0;
    ptPkt->tHead.ulDestId = 0;
    ptPkt->tHead.ulDest = 0x20; /* addressed communication channel */
    ptPkt->tData.ulParameterID = PID_ECS_DEVICE_IDENTIFICATION;
    ptPkt->tData.ulParameterLength = 4;
    ptPkt->tData.abParameter[0] = usIdentValue & 0xFF;
    ptPkt->tData.abParameter[1] = usIdentValue >> 8;
    ptPkt->tData.abParameter[2] = 0;
    ptPkt->tData.abParameter[3] = 0;
}
```

5.3.3 Legacy behavior

NOTE This legacy handling mechanism is allowed, only for backward-compatibility reasons for existing slave devices before the Explicit Device Identification was specified. The feature should not be used for new developments.

The legacy mode according to ETG1020 [11] defines the behaviour concerning a simultaneous use of the address that is set locally on the device e.g. via rotary switch or display and the Configured Station Alias coming from MainDevcie. This can lead to a race condition in some chips. Even if there is no race condition possible by design in netX90, some devices want to support the legacy mode for chip replacement with minimum differences.

In short: The legacy behavior applies when both addresses are set with different values, not zero as well while the device is started. In this case the rotary switch value is copied to register 0x12 of the ESC and Configured Station Alias is set to 0. An CoE Emergency is generated when a change to SafeOP is requested. After going back to state Init or a restart, the SubDevice can be brought to state Operational again. See [11] or ETG's Knowledge Base website. which provides details.

5.3.3.1 How to implement the legacy mode support

The following description is for firmwares with minimum version number V5.0.0.0.

- Read the local address value of your device.
- Set up the ECAT_SET_CONFIG_REQ adding the component ECAT_SET_CONFIG_UID and set the parameter usDeviceIdentificationValue = local address value. (The parameter usStationAlias = 0, it has no effect here.)
- If the local address is **not zero** send the *ECAT_DPM_SET_STATION_ALIAS_REQ** with parameter usStationAlias = local address value. If the local address is zero, you do not send the station alias request and no legacy handling applies as specified in ETG.1020.
- The former step activates the legacy support in the firmware and must be done **before switching the bus on** with HIL_START_STOP_COMM_REQ.
- ESI: For legacy support you need the following entries in the ESI <Info> tag
 - <IdentificationAdo>#x12</IdentificationAdo> .
 - <IdentificationReg134>1</IdentificationReg134>
- The mechanism for updating the local address value via HIL_SET_FW_PARAMETER_REQ remains unchanged. From now on, this update must be performed every time the local address changes.

5.3.3.2 How to test the legacy mode

Testing the mechanism with TwinCAT can be challenging. Using the Conformance Test Tool is generally more reliable. However, if you need to use TwinCAT, the key steps to ensure success are outlined below.

Prerequisites:

- Use a second SubDevice in the line before your DUT. This means that the master will not detect a complete network loss in case of the power cycle in the further steps. Otherwise, TwinCAT will exhibit strange behaviour in the subsequent test steps.
- Uncheck the following state machine settings in Advanced Settings > General > Behavior: 'Auto Restore States' and 'Relnit after Communication Error'. If you also select 'Init' as the target state, it is easier to see the subsequent steps.
- Click Reload to apply the changes.

Test Procedure:

- After writing 'New Value' with the 'Write to EEPROM' button in the Advanced settings for Configured Station Alias, you will see the value in the 'Actual Value (EEPROM)' field. The next step is to power cycle the DUT.
- The device will come up with the former EEPROM value in Init (if you have chosen Init as the target state, you can check this), as well as the ID selector value in register 0x12.
- Switching the state to PreOP forces the firmware to set the EEPROM value to zero. If the TwinCAT online surface hangs and does not switch the slave to PreOP, press the Reload button. (This is not a SubDevice problem, as you can verify by reading the registers with the 'Advanced settings' option).
- When requesting SafeOP the Emergency error message with AL Status Code 0x0061 appears, as expected by the specification. Switching to Init allows you to proceed normally.

HINT: These steps are only valid if the application does not handle the SII content or the remanent data itself (tag list setting). These are rare cases and we cannot guarantee that it is possible to implement legacy mode at all.

5.4 Complete Access for object data held by application

This section describes the sequence of packets for the use case "object dictionary holds object and subobjects descriptions, application holds data" for a Complete Access. For a description of the use case, see reference [4].

This section is not for the use case dynamic PDO mapping, therefore see [Section 5.5](#).

The ODV3 task translates SDO up- and downloads with Complete Access from an EtherCAT MainDevice into single accesses to the application. The main difference of the Complete Access compared to the standard access is the order when the object validation indications are sent: with Complete Access, the validation indications are implicit.

If a Complete Access request from EtherCAT MainDevice fails, the application has to restore the former object values.

The following example shows how to handle the packets for a write object request correctly to support both single access and Complete Access.