Migration Guide

PROFINET IO Device

Migration from V3.4 to V3.5

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

Table of Contents

1 Introduction

1.1 About this Document

This document describes the steps required to migrate an existing PROFINET IO Device application from stack/firmware Version V3.4 to V3.5

1.2 List of Revisions

Rev	Date	Name	Chapter	Revision
1	2012-06-25	BM/HH	all	Created
2	2012-10-31	BM	All	Corrected (sub)module IDs, editorial changes Added PNS_IF_LINK_STATUS_CHANGE_IND to services that changed

Table 1: List of Revisions

1.3 References to Documents

This document refers to the following documents:

[1] Hilscher Gesellschaft für Systemautomation mbH: Protocol API, PROFINET IO RT/IRT Device, V3.5.x.x, Revision 3, English, 2012.

Table 2: References to Documents

1.4 Legal Notes

1.4.1 Copyright

© Hilscher, 2012, Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

1.4.2 Important Notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

1.4.3 Exclusion of Liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

1.4.4 Export

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different counters, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

2 General

The PROFINET IO Device Firmware/Stack V3.5 supports *Shared Device* as a new feature. Firmware/Stack V3.5 supports 2 IO ARs at the same time. Therefore an application must be able to deal with requests of multiple application relations (AR) at the same time. For packet communication the field `hDeviceHandle` is used by the firmware/stack to indicate the associated Application Relation.

Multiple ARs require some handling of ownership. Only one AR is allowed to write data to an output, otherwise inconsistent data might be given to a submodule. This ownership handling is done inside the PROFINET IO Device stack and completely hidden from the user.

Special care needs to be taken when handling PNSIF_CHECK_IND packets. These packets indicating a difference between local submodule configuration and expected submodule configuration of IO Controller. In contrast, firmware/Stack V3.4 only generated this packets during connection establishment. Only one application relation was supported by firmware/stack V3.4. There was only a small timeslot when this indication was created. As a result of the feature shared device these indications are now additionally generated in different situations.

- One AR exists, another AR is established. A submodule not locally plugged is requested by the second AR. Now a check indication is sent to user application although there is an AR already established.

- One AR exists, another AR is established. First AR expects output submodule in slot 1 subslot 1. Second AR expects a different output submodule in slot 1 subslot 1. At first the second IO Controller will be informed that the submodule can not be used and no check indication is generated. If now the first AR breaks down the firmware/stack will generate a check indication to the user application and inform the user about the difference between expected and real configuration. This check indication is generated completely out of context of AR establishment.

The cyclic data exchange with multiple application relations is handled by the firmware/stack V3.5 internally in the following ways (according PROFINET specification):

- For outputs only the data of the submodules owner AR is copied into the dual-port memory input area/consumer image

- For inputs the data is copied from the dual-port memory output area/provider image to all ARs requesting the submodule. This is called *shared input*.

The parameter records access is also controlled by the owner ship:

- Parameter writes (`PNS_IF_WRITE_RECORD_REQ`) are only allowed from the submodules owner AR. Requests from other ARs will be filtered by firmware/stack automatically.

- Parameter reads (`PNS_IF_READ_RECORD_REQ`) are allowed from all ARs.

2.1 IRT

The PROFINET IO Device firmware/stack V3.5 uses a new IRT technology called *Relative Forwarder* whereas the firmware/stack V3.4 implemented an *Absolute Forwarder*. Therefore, if the application uses IRT, the GSDML file must be adapted.

Important: If an existing IRT capable device is upgraded from a firmware/stack V3.4 based firmware to a firmware/stack V3.5 based firmware, the PROFINET IO Controllers configuration must be updated using the new GSDML file.

---

3 GSDML Device Description

As stated above, if the application supports IRT, the GSDML file must be updated to Version 2.3 because older GSDML versions do not support the *Relative Forwarder`* technology. Besides that the new Stack supports multiple ARs (*Shared Device*). Both features require, that the following parameters should be used for the interface submodule

```
<InterfaceSubmoduleItem ID="DIM 5 Interfacesubmodule"
    SubslotNumber="32768" SubmoduleIdentNumber="0x00002001"
    DCP_HelloSupported="true" NetworkComponentDiagnosisSupported="true"
    SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" IsochroneModeInRT_Classes="RT_CLASS_3"
    SupportedMibs="MIB2" SupportedProtocols="SNMP;LLDP"
    TextId="PN-IO">

  <RT_Class3Properties ForwardingMode="Relative" MaxBridgeDelay="3460"
    MaxNumberIR_FrameData="1024" StartupMode="Legacy"/>

  <SynchronisationMode SupportedSyncProtocols="PTCP" SupportedRole="SyncSlave"
    MaxLocalJitter="50"  T_PLL_MAX="1000"/>

  <ApplicationRelations NumberOfAR="2" StartupMode="Advanced">
    <TimingProperties SendClock="32 64 128"
      ReductionRatio="1 2 4 8 16 32 64 128 256 512"/>
    <RT_Class3TimingProperties SendClock="8 16 32 64 128"
      ReductionRatio="1 2 4 8 16"/>
  </ApplicationRelations>
  <MediaRedundancy SupportedRole="Client"/>
</InterfaceSubmoduleItem>
```

As a further consequence of this new IRT technology, Hilscher has assigned new module and submodule ident numbers to the device access point. If the stack is configured using the netX Configuration Tool, the following numbers shall be used by the application to match Hilscher default values for cifX:

Description	Value for V3.4	Value for V3.5
DAP ModuleIdentNumber	0x00001001 or 0x00001101	0x00002001
DAP SubmoduleIdentNumber	0x00001000 or 0x00001100	0x00002000
Interface SubmoduleIdentNumber	0x00001001 or 0x00001101	0x00002001
Port 1 SubmoduleIdentNumber	0x00001002 or 0x00001102	0x00002002
Port 2 SubmoduleIdentNumber	0x00001003 or 0x00001103	0x00002003

Table 3: Module/Submodule Ident Number Comparison Firmware/Stack V3.4 and V3.5

Usage of other values is possible as well.

4   Packet Handling

As already stated, the protocol firmware/stack V3.5 now supports multiple ARs at the same time. Each AR is assigned a specific device handle (hDeviceHandle) which may be used to relate indications and requests with a specific AR. One import consequence of this new feature is, that the application must be able to deal with multiple indications of the same type at the same time. This comes into account if the application stores a packet for a time before returning it back to the stack. The following services are affected by this behavior:

Affected Service	Description
PNS_IF_PARAM_END_IND / PNS_IF_PARAM_END_RES	Parameter End Service
PNS_IF_SET_APPL_READY_REQ / PNS_IF_SET_APPL_READY_CNF	Application Ready Service
PNS_IF_AR_CHECK_IND / PNS_IF_AR_CHECK_RES	AR Check Service
PNS_IF_AR_INDATA_IND / PNS_IF_AR_INDATA_RES	AR InData Service
PNS_IF_AR_ABORT_IND / PNS_IF_AR_ABORT_RES	AR Abort Service
PNS_IF_APDU_STATUS_IND / PNS_IF_APDU_STATUS_RES	APDU Status Service
PNS_IF_ALARM_IND / PNS_IF_ALARM_RES	Alarm Service
PNS_IF_READ_IM_IND / PNS_IF_READ_IM_RES	Read Identification & Maintenance Service
PNS_IF_WRITE_IM_IND / PNS_IF_WRITE_IM_RES	Write Identification & Maintenance Service
PNS_IF_READ_RECORD_IND / PNS_IF_READ_RECORD_RES	Read Record Service
PNS_IF_WRITE_RECORD_IND / PNS_IF_WRITE_RECORD_RES	Write Record Service
PNS_IF_PARAM_BEGIN_IND / PNS_IF_PARAM_BEGIN_RES	Parameter Begin Service
PNS_IF_CONNECT_REQ_DONE_IND / PNS_IF_CONNECT_REQ_DONE_RES	Connect Request Done Service
PNS_IF_RELEASE_RECV_IND / PNS_IF_RELEASE_RECV_RES	Release Received Service
PNS_IF_SR_INFO_IND / PNS_IF_SR_INFO_RES	System Redundancy Info Service

Table 4: Packets/Services affected by hDeviceHandle

The stacks V3.5 packet API has been extended by the following services:

New Service	Description
PNS_IF_SR_INFO_IND / PNS_IF_SR_INFO_RSP	Indicates changes of the System Redundancy State

Table 5: New Packets/Services

Furthermore the following services have been changed:

Changed Service	Description
PNS_IF_SET_CONFIGURATION_REQ / PNS_IF_SET_CONFIGURATION_CNF	Set Configuration Service
PNS_IF_SET_IOXS_CONFIG_REQ / PNS_IF_SET_IOXS_CONFIG_CNF	Set IOxS Config Service
PNS_IF_SET_OEM_PARAMETERS_REQ / PNS_IF_SET_OEM_PARAMETERS_CNF	Set OEM Parameters Service
PNS_IF_LINK_STATE_CHANGE_IND / PNS_IF_LINK_STATE_CHANGE_RES	Link state change indication replaced with generic rcX packet RCX_LINK_STATUS_CHANGE_IND / RCX_LINK_STATUS_CHANGE_RES

Table 6: Changes Packets/Services

5   Interface dependent Changes

5.1   Dual-Port Memory

As a consequence of the Shared Device feature, the COMMUNICATING-Bit is handled as follows:

- ■   The COMMUNICATING-Bit is set when the first AR enters valid Data-Exchange
- ■   The COMMUNICATING-Bit is cleared when the last AR is shut down

5.2   Linkable Object Module

The internal structure of stack V3.5 has changed in many ways so that several changes are required when using the *Linkable Object Module* (LOM). At first, the following versions of additional Components are required:

Component	Version
rcX Operating System	2.0.8.8 or greater
TCP/IP Stack	2.1.11.0 or greater
Shared Memory (Optional)	0.990 or greater

Table 7: Required Components for Linkable Object Module

5.3   Stack Configuration and Initialization

Stack V3.5 uses a new initialization and startup sequence. The stack is not started using the *Static Task List* but by invoking a single function from the application. This simplifies starting the stack greatly.

Note:	If the shared memory API is used the PROFINET IO Device Stacks AP task will start the stack.

In order to adapt to the new stack, all PROFINET protocol stack tasks shall be removed from the static tasks list at first. Furthermore, the following hardware settings need to be changed. For a full list of the recommended rcX configuration please see document reference [1].

Stack V3.5 uses the RT-TIMER in continuous mode. This simplifies the design greatly. As a consequence the time interval of the RT-Timer determines the minimum RT send clock:

```
STATIC RX_HWTIMER_SET_T atrXHwTim[] = {
    /* ... */
    {{PNS_TIMER_IRQ_NAME,RX_PERIPHERAL_TYPE_TIMER,0}, /* System Timer */
     1,                                     /* use GPIO_counter0 */
     1000,                                  /* 1000 microseconds = 1msec */
     TRUE,                                  /* Continuous Mode */
     TRUE,                                  /* Interrupt enabled */
     FALSE,                                 /* No external Clock */
     RX_HWTIMER_TRIGGER_NONE,               /* No Trigger */
     0,                                     /* No I/O reference */
     0                                      /* No Prescaler */
    },
    /* ... */
};
```

If the IRT switch is used, the parameters of the EDD driver need to be adapted as well. The following settings are recommended:

```
STATIC RX_EDD_PARAMETERS_T g_atEddParam[]= {
    { RX_EDD_PARAM_PHY_NAME,        PNS_PHY0_NAME, 0},
    { RX_EDD_PARAM_PHY1_NAME,       PNS_PHY1_NAME, 1},

    { RX_EDD_PARAM_XPEC_NAME,       "PNS_XPEC",     0 },
    { RX_EDD_PARAM_XMAC_RPU_NAME,   "PNS_XMACRPU",  0 },
    { RX_EDD_PARAM_XMAC_TPU_NAME,   "PNS_XMACTPU",  0 },
    { RX_EDD_PARAM_INTERRUPT_NAME,  "PNS_COM",      0 },
    { RX_EDD_PARAM_FIFO_NAME,       "PNS_FIFO",     0 },

    { RX_EDD_PARAM_XPEC1_NAME,      "PNS_XPEC",     1 },
    { RX_EDD_PARAM_XMAC1_RPU_NAME,  "PNS_XMACRPU",  1 },
    { RX_EDD_PARAM_XMAC1_TPU_NAME,  "PNS_XMACTPU",  1 },
    { RX_EDD_PARAM_INTERRUPT1_NAME, "PNS_COM",      1 },
    { RX_EDD_PARAM_FIFO1_NAME,      "PNS_FIFO",     1 },
#if defined (__NETX500) || defined (__NETX100)
    { RX_EDD_PARAM_XPEC3_NAME,      "PNS_XPEC",     3 },
    { RX_EDD_PARAM_XMAC3_RPU_NAME,  "PNS_XMACRPU",  3 },
    { RX_EDD_PARAM_XMAC3_TPU_NAME,  "PNS_XMACTPU",  3 },
    { RX_EDD_PARAM_INTERRUPT3_NAME, "PNS_COM",      3 },
    { RX_EDD_PARAM_FIFO3_NAME,      "PNS_FIFO",     3 },
#endif
    { RX_PNIRT_EDD_PARAM_MSYNC0_IRQ_NAME, PNS_MSYNC_IRQ_NAME,0 },
    { RX_PNIRT_EDD_PARAM_MSYNC1_IRQ_NAME, PNS_MSYNC_IRQ_NAME,1 },

    { RX_PNIRT_EDD_PARAM_COMM_TASK_NAME,     "PNS_IRQ_TSK", 0},
    { RX_PNIRT_EDD_PARAM_COMM_TASK_PRIO, (void*)TSK_PRIO_4, 0},
    { RX_PNIRT_EDD_PARAM_COMM_TASK_TOK,  (void*)TSK_TOK_4, 0},

    { RX_PNIRT_EDD_PARAM_COMPUTE_TASK_NAME,    "PNS_COMP_TSK", 0},
    { RX_PNIRT_EDD_PARAM_COMPUTE_TASK_PRIO, (void*)TSK_PRIO_14, 0},
    { RX_PNIRT_EDD_PARAM_COMPUTE_TASK_TOK,  (void*)TSK_TOK_14, 0},
    { RX_EDD_PARAM_END_OF_LIST}
};
```

If the shared memory API is used, the PNS_AP task shall be added to the static tasks list as follows:

```
PNS_AP_DPM_STARTUPPARAMETER_V3_T g_tPnsApStartupPrm =
{
    TLR_TASK_PNIO_APDEV,
    PNS_AP_DPM_STARTUPPARAMETER_VERSION_3,
    0,                          /* Instance */
                                /* Flags (Set to zero for non IRT))*/
    (PNS_AP_DPM_STARTUP_FLAG_USE_IRT | PNS_AP_DPM_STARTUP_FLAG_SYNC_APPL_WITH_IRQ),
    8,                          /* MinDeviceInterval */
    2,                          /* Ethernet Port counter - usually we use 2 Ports */
    0,                          /* DPM Channel Number */
    PNS_EDD_IDENTIFY_NAME,    /* Identify Name of the Ethernet Driver */
    PNS_MSYNC_IRQ_NAME,       /* Identify Name of the Motion Sync 0 Interrupt (For IRT
ONLY)*/
    PNS_MSYNC_IRQ_NAME,       /* Identify Name of the Motion Sync 1 Interrupt (For IRT
ONLY)*/
    PNS_RT_IRQ_NAME,          /* Identify Name of interrupt associated with the RT Timer
*/
    "PNIO_SYNC_IRQ",          /* Identify Name of the Sync Interrupt (This is the
NS_COMPARE_IRQ) *)
    PNS_TIMER_IRQ_NAME,       /* Identify Name of the RT Timer (Used for RT Mode) */
    "PNS_SF",                 /* SF LED */
    "PNS_BF",                 /* BF LED */
    "PNS_SF",                 /* Blink LED */
    NULL,                     /* Flash parameters Set this if the AP-TASK shall use the
Flash*/
    0,                        /* EDD Instance */
    0,                        /* MSync0 IRQ Instance */
    1,                        /* MSync1 IRQ Instance */
```

```
    0,                          /* RT Scheduler IRQ Instance */
    0,                          /* Sync IRQ Instance */
    0,                          /* SF LED Instance */
    0,                          /* BF LED Instance */
    0,                          /* Blink LED Instance */
                                /* Task Priority Settings */
    .tTaskRtc   = {NULL, "PNIO_RTC", TSK_PRIO_10},
    .tTaskCmdev = {NULL, "PNIO_CMDEV", TSK_PRIO_18},
    .tTaskRta   = {NULL, "PNIO_RTA", TSK_PRIO_13},
    .tTaskLldp  = {NULL, "LLDP-Task", TSK_PRIO_19},
    .tTaskMibDB = {NULL, "Mib-Database", TSK_PRIO_23},
    .tTaskPnsif = {NULL, "PNS_IF", TSK_PRIO_21},
    .tTaskRpc   = {NULL, "RPC", TSK_PRIO_16},
    .tTaskSnmp  = {NULL, "SNMP-Server", TSK_PRIO_22},
    0x011E,                     /* Default Profinet Vendor ID to use */
    0x0103,                     /* Default Profinet Device ID to use */
    .patPNSModules = NULL,   /* Points to list of extension modules to hook into the
Stack */
};

RX_STATIC_TASK_T g_atStaticTasks[] = {
    /* ... */
    {"PNS_AP",                               /* Set Identification */
     TSK_PRIO_40, TSK_TOK_40,                /* Set Priority,and Token ID */
     0,                                      /* Set Instance to 0 */
     0,                                      /* Pointer to Stack */
     1024,                                   /* Size of Task Stack */
     0,                                      /* Threshold to maximum possible value */
     RX_TASK_AUTO_START,                     /* Start task automatically */
     (void FAR*)TaskEnter_PnsAp,             /* Task function to schedule */
     NULL,                                   /* Function called when Task will be
deleted */
     (UINT32)&g_tPnsApStartupPrm,            /* Startup Parameter */
     {0,0,0,0,0,0,0,0}                       /* Reserved Region */
    },
    /* ... */
};
```

If the application uses packet interface with the stack, the application is responsible for starting the stack. Therefore initialization sequence of the application shall be extended as follows:

```
TLR_HANDLE                               hQuePnsIf;  /* PNS-IF Task Queue */
PROFINET_IODEVICE_TASK_RESOURCES_T *ptPNSRsc   /* Pointer to PNS Resources */

{ /* Start The stack */
    PROFINET_IODEVICE_STARTUPPARAMETER_T tPNSParam /* PNS Stack Parameters */

    tPNSParam.pszEddName          = PNS_EDD_IDENTIFY_NAME;
    tPNSParam.pszHwTimRTSchedName = PNS_TIMER_IRQ_NAME;
    tPNSParam.pszIrqMSync0Name    = PNS_MSYNC_IRQ_NAME;
    tPNSParam.pszIrqMSync1Name    = PNS_MSYNC_IRQ_NAME;
    tPNSParam.pszIrqRTSchedName   = PNS_RT_IRQ_NAME
    tPNSParam.pszIrqSyncApplName  = "PNIO_SYNC_IRQ";
    tPNSParam.ulEthPortCnt        = 2;
    tPNSParam.ulMaxAr             = 2;
    tPNSParam.ulFlags             =
        PROFINET_IODEVICE_STARTUP_FLAG_USE_IRT |
        PROFINET_IODEVICE_STARTUP_FLAG_SYNC_APPL_WITH_IRQ;
    tPNSParam.ulInstance          = 0;

    /* configure the priorities of all tasks */
    tPNSParam.tTaskRtc   = {NULL, "PNIO_RTC", TSK_PRIO_10},
    tPNSParam.tTaskCmdev = {NULL, "PNIO_CMDEV", TSK_PRIO_18},
    tPNSParam.tTaskRta   = {NULL, "PNIO_RTA", TSK_PRIO_13},
    tPNSParam.tTaskLldp  = {NULL, "LLDP-Task", TSK_PRIO_19},
    tPNSParam.tTaskMibDB = {NULL, "Mib-Database", TSK_PRIO_23},
    tPNSParam.tTaskPnsif = {NULL, "PNS_IF", TSK_PRIO_21},
    tPNSParam.tTaskRpc   = {NULL, "RPC", TSK_PRIO_16},
    tPNSParam.tTaskSnmp  = {NULL, "SNMP-Server", TSK_PRIO_22},
```

```
    /* now start the stack */
    If (TLR_S_OK == PNS_StackInit(&tPNSParam, &ptPNSRsc))
    {
      hQuePnsIf = ptPNSRsc->hQuePnsif;
    }
}
```

Note: The handle `hQuePnsIf` is the handle to the protocol stacks message queue.

6   Appendix

6.1   List of Tables

6.2 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com
Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn
Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr
Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
New Delhi - 110 065
Phone: +91 11 43055431
E-Mail: info@hilscher.in

Italy
Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it
Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp
Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Suwon, Gyeonggi, 443-734
Phone: +82 (0) 31-695-5515
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch
Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us
Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com