IEC 62769-6

Edition 1.0 2015-05

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field Device Integration (FDI) – Part 6: FDI Technology Mapping

Intégration des appareils de terrain (FDI) – Partie 6: Mapping de technologies FDI

IEC 62769-6:2015-05(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue
Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,…). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 15 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary
Plus de 60 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 62769-6

Edition 1.0 2015-05

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field Device Integration (FDI) – Part 6: FDI Technology Mapping

Intégration des appareils de terrain (FDI) – Partie 6: Mapping de technologies FDI

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040.40; 35.100

ISBN 978-2-8322-2635-3

Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI) –

Part 6: FDI Technology Mapping

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

International Standard IEC 62769-6 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this standard is based on the following documents:

CDV	Report on voting
65E/349/CDV	65E/426/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 62769 series, published under the general title *Field Device Integration (FDI)*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning

a) Method for the Supplying and Installation of Device-Specific Functionalities, see Patent Family DE10357276;

b) Method and device for accessing a functional module of automation system, see Patent Family EP2182418;

c) Methods and apparatus to reduce memory requirements for process control system software applications, see Patent Family US2013232186;

d) Extensible Device Object Model, see Patent Family US12/893,680.

IEC takes no position concerning the evidence, validity and scope of this patent right.

The holders of these patent rights have assured the IEC that he/she is willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC. Information may be obtained from:

a) ABB Research Ltd
Claes Rytoft
Affolterstrasse 4
Zurich, 8050
Switzerland

b) Phoenix Contact GmbH & Co KG
Intellectual Property, Licenses & Standards
Flachsmarktstrasse 8, 32825 Blomberg
Germany

c) Fisher Controls International LLC
John Dilger, Emerson Process Management LLLP
301 S. 1st Avenue, Marshaltown, Iowa 50158
USA

d) Rockwell Automation Technologies, Inc.
1 Allen-Bradley Drive
Mayfield Heights, Ohio 44124
USA

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line data bases of patents relevant to their standards. Users are encouraged to consult the data bases for the most up to date information concerning patents.

FIELD DEVICE INTEGRATION (FDI) –

Part 6: FDI Technology Mapping

1 Scope

This part of IEC 62769 specifies the technology mapping for the concepts described in the Field Device Integration (FDI) standard. The technology mapping focuses on implementation regarding the components FDI Client and User Interface Plug-in (UIP) that are specific only to the workstation platform as defined in IEC 62769-4:2015, Annex E.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62541 (all parts), *OPC Unified Architecture*

IEC 61804 (all parts), *Function blocks (FB) for process control*

IEC 62769-1, *Field Device Integration (FDI) – Part 1: Overview*

NOTE IEC 62769-1 is technically identical to FDI-2021.

IEC 62769-2, *Field Device Integration (FDI) – Part 2: FDI Client*

NOTE 1 IEC 62769-2 is technically identical to FDI-2022.

NOTE 2 IEC 62769-2 is technically identical to FDI-2023.

IEC 62769-4:2015, *Field Device Integration (FDI) – Part 4: FDI Packages*

NOTE IEC 62769-4 is technically identical to FDI-2024.

IEC 62769-5, *Field Device Integration (FDI) – Part 5: FDI Information Model*

NOTE 1 IEC 62769-5 is technically identical to FDI-2025.

NOTE 2 IEC 62769-5 is technically identical to FDI-2027.

ISO/IEC 19505-1, *Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure*

ISO/IEC 29500, (all parts) *Information technology – Document description and processing languages – Office Open XML File Formats*

3 Terms, definitions, abbreviated terms, acronyms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62769-1 as well as the following apply.

3.1.1
Application Domain
isolated environment where applications execute

3.1.2
Assembly
reusable, version information providing, and self-describing building block of a CLR application

Note 1 to entry: This note applies to the French language only.

3.1.3
FDI Type Library
assembly that contains the interfaces and data types that are used for the data exchange and interaction between a UIP and an FDI Client

Note 1 to entry: This note applies to the French language only.

Note 2 to entry: This note applies to the French language only.

3.1.4
Global Assembly Cache
machine-wide code cache that stores Assemblies specifically designated to be shared by several applications

3.1.5
Windows Registry
system-defined database in which applications and system components store and retrieve configuration data

3.2 Abbreviated terms and acronyms

For the purposes of this document, the abbreviated terms and acronyms given in IEC 62769-1 as well as the following apply.

CLR	Common Language Run-time
MSI	Microsoft Installer
WPF	Windows Presentation Foundation
UML	Unified Modeling Language

3.3 Symbols

Figures in this document use the graphical symbols according to ISO/IEC 19505 (UML 2.0).

4 Technical concepts

4.1 General

4.1.1 Overview

In 4.1.2, 4.2, 4.3, 4.4, and 4.5, this document describes first the technology base for UIP implementation, the hardware and software environment including the related implementation rules. Clause 4 follows a life cycle (use case) oriented approach.

Subclause 4.6 describes the copy deployment procedures and related implementation rules for the UIP and the FDI Client.

UIP executable instantiation and termination is described in 4.7.

Subclause 4.8 defines the rules about interaction between the FDI Client and the UIP.

Security related definitions are written in 4.9.

The service interface definitions for the FDI Client and the UIP are found in Clause 5.

4.1.2 Platforms

The UIP and FDI Client shall be built upon the Microsoft .NET Framework and executed in the .NET Common Language Run-time.

The minimum set of workstation supported I/O devices is: mouse, keyboard, and color screen resolution of 1024 x 768 pixels.

The following Table 1 lists all the technologies and their editions that are consistent with FDI components.

Table 1 – Technology edition reference

Technology	Standard	Edition
.NET	N/A	CLR4 for UIP Implementation
EDDL	IEC 61804	2014
OPC UA (Parts 1-8)	IEC 62541	2015 (to be published)
Open Packaging Convention	ISO/IEC 29500	2011
Extensible Markup Language (XML)	N/A	W3C, 1.0 (fifth edition)

4.1.3 FDI Type Library

The Device Access Services and the UIP Services can be modeled as .NET interfaces passing .NET data type arguments. These interfaces and data types are used for the data exchange and interaction between the UIP and the FDI Client. For runtime error handling purposes during interface method calls .NET exceptions classes are defined.

The FDI .NET interfaces, data types, and exception classes are defined in a single FDI Type Library. The FDI Type Library is a strong named Assembly. The FDI Type Library is signed with a single unique key. The FDI Type Library shall be installed as part of the FDI Client installation and not with a UIP.

FDI Type Libraries shall not be registered within the Global Assembly Cache.

The FDI Client shall install FDI Library Versions for all Technology Versions that it supports.

The FDI Type Library shall be installed in such way that it is shared between the UIP and the FDI Client.

Figure 1 shows the FDI Type Library structure.

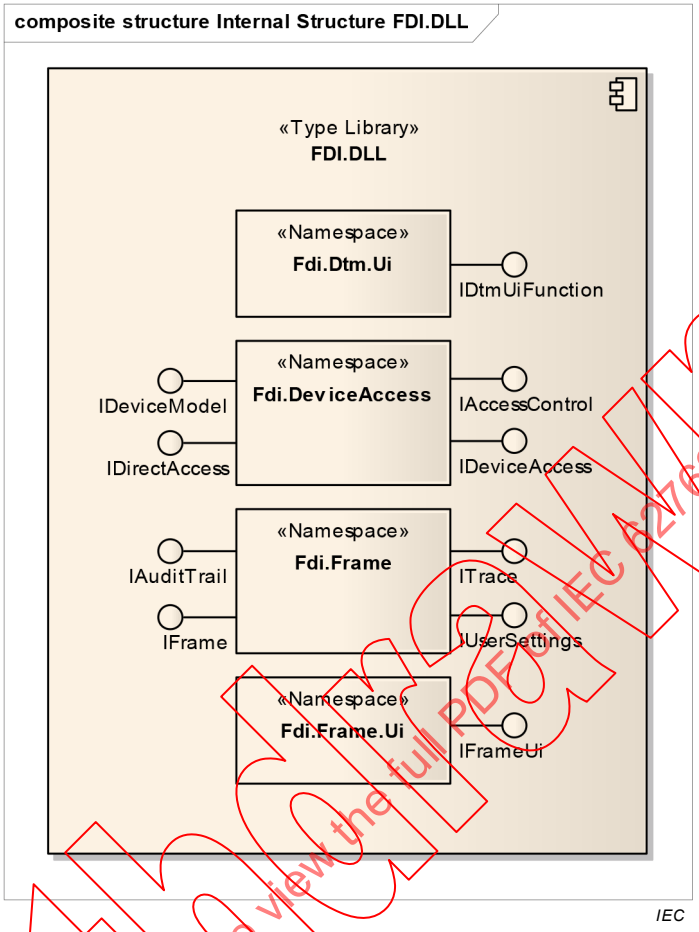


Figure 1 – FDI Type Library structure

NOTE The composite structure diagram shows only the core interfaces that implement the interfaces defined in IEC 62769-2.

4.2 UIP representation

The UIP Variant can contain either a single or multiple runtime modules (.NET Assembly) and their related supplementary files (for example: resource files). The runtime module of the IP Variant is called UIP executable. The supplementary file(s) of the UIP Variant is/are called UIP supplement(s).

UIP supplement(s) is/are stored under (a) subfolder(s) of the UIP executable installation directory

EXAMPLE Examples of UIP supplementary data files include resource files and application configuration data.

The RuntimeId of a UIP Variant shall be ".NET Framework CLR4", see IEC 62769-4.

The UIP Variant shall be self-contained. All UIP required libraries (.NET Assemblies) required by a UIP Variant are stored within the same Folder.

4.3 UIP executable representation

The implementation of the UIP depends on the type of user interface elements that can be embedded into the user interface hosting environment of the FDI Client. UIP shall be

implemented as a .NET `System.Windows.Forms` class `UserControl` or a Windows Presentation Foundation (WPF) `System.Windows.Controls` class `UserControl`.

UIP executables and their required libraries shall have strong names. The signing of a strong named Assembly can be done using a self-generated key.

NOTE The identity of strong named Assemblies consists of a name, version, culture, public key token and digital signature.

UIP executables and their required libraries shall be shipped with file containing the public key in order to enable Assembly verification.

4.4 UIP executable compatibility rules

The UIP component provided version information consists of:

<Major>.<Minor>.<Build Number>.<Revision>

UIP components using the same identity (UipId/IEC 62769-5) that are showing a different value in position <Major> are not compatible with each other. Any other difference showed in the version information between the same UIP component identities means that those UIP component identities are compatible. A newer UIP component is allowed to overwrite an older UIP component without breaking the intended functionality.

The compilation target platform for the UIP shall be “anyCPU”. If this is not feasible the UIP shall be shipped in two variants. One UIP variant shall be compiled for target platform “x86”. The second UIP variant shall be compiled for target platform “x64”. The compilation platform target shall be described in the catalog.xml file which is defined in IEC 62769-4. This catalog.xml file contains an xml element “CpuInformation” that describes the User Interface Plug-in variant. The allowed values that shall be used in the xml element “CpuInformation” are “anyCPU”, “x86” or “x64”.

4.5 Allowed .NET Common Language Run-time versions

4.5.1 General

Specific CLR (Common Language Run-time) versions are released for the execution of software components built with specific .NET Framework versions. The .NET CLR version 4.0 is used to execute software components built with .NET Framework 4.0. .NET Components are built for one CLR version only but can be capable to run also under a newer CLR version.

FDI Clients can be built based on CLR version 4.0 or future versions. An FDI Client has to realize the following situations when starting a UIP.

- When the UIP to be started was built for the same run-time, the UIP can be started in the FDI Client as usual.
- When the UIP to be started was built with another CLR version and is not compiled for the current running CLR version, the FDI Client shall start the UIP in a surrogate process with the adequate CLR version. (More details are described in 4.5.2.)

Taking this behavior in account, a UIP shall be developed for CLR version 4.0 or any future version. In case the CLR versions do not match, the UIP shall be started in a separate process. The UIP will then not be displayed as an integrated module within the FDI Client. It is up to the FDI Client to realize the surrogate process.

4.5.2 CLR compatibility strategy

In the future, FDI Clients and UIPs will be permitted to be built on different incompatible versions of the CLR.

If an FDI Client detects that a UIP requires a CLR that is not compatible with the FDI Client, the FDI Client can use a proxy class that enables interaction with the UIP built using a different version of the CLR.

The FDI Client loads a proxy UIP executable, creates an instance of the proxy class, and delegates the execution of the UIP to this proxy. The proxy starts a process with the required CLR and executes the UIP in this surrogate process. The proxy classes provide the standard FDI interfaces. The FDI Client can use these interfaces to interact with the UIP executed in the surrogate process.

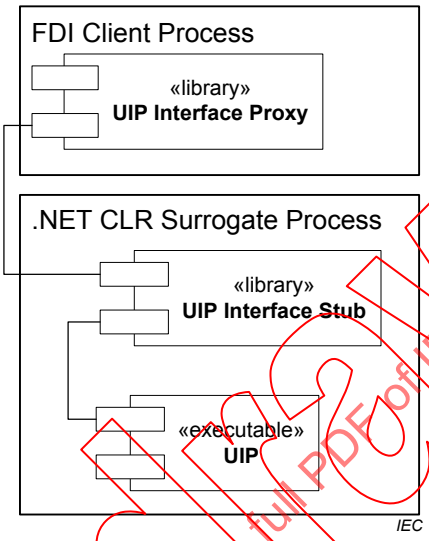


Figure 2 – .NET surrogate process

4.5.3 How to identify the .NET target platform of a UIP

The .NET target platform CLR version information for which a certain Assembly is compiled can be extracted by means of .NET Framework library functions (see Figure 3).

```
clrVersion = Assembly.LoadFrom(<Assembly Path>).ImageRuntimeVersion;
```

IEC

Figure 3 – Identification of Run-time Version

NOTE The Visual Studio[1] 2008 and 2010 IDE allow developers to select the .NET Framework target. The selection of a .NET Framework target older than the base for the current Visual Studio IDE automatically creates a configuration file listed as “app.config” within the solution explorer. This file only reflects the current complier setting. The compiler does not read that file.

4.6 Installing UIP

The FDI Server imports the UIP from an FDI Package.

The UIP installation is done per file copy only. The UIP executable shall not be registered within the Global Assembly Cache. The UIP is installed within a folder structure, which is

[1] Visual Studio is the trade name of Microsoft Corporation. This information is given for the convenience of users of this part of IEC 62769 and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

called the UIP folder structure. The FDI Client shall manage the UIP folder structure. The UIP folder structure shall separate the UIP Variants from each other in order to avoid file name conflicts. UIP executables shall be installed to a path that allows browse read and write access.

Since the FDI Client manages the folder structure the UIP shall not perform any access to an absolute path. Any file access shall be done relative to the installation root of the UIP.

According the version management described in IEC 62769-4, the coexistence of major version changes of UIP of the same type shall be supported. This shall be done by installing a newer UIP into a separate folder. The “strong-name” rule ensures that related Assemblies can coexist during runtime.

The FDI Client implementation ensures that UIP deployment works independently from current user credentials. (See the NOTE below.)

NOTE Certain operating system managed folders require specific access rights, for example, modifications in folder “Program Files” require “Administrator” rights. The Windows operating system provides several means to allow an application running with restricted user rights, to execute actions with administrator privileges transparent to the user, for example, special restriction handling for identified directories, services with administration rights, executables that are configured to automatically run with administration rights. The alternative is to copy UIP executables into folders writeable for “normal” users.

4.7 UIP Lifecycle

4.7.1 General

The UIP state machine, outlined in IEC 62769-4, is composed of the Loaded, Created, Operational, Deactivated and Disposed states. The mechanisms affecting state changes are described in 4.7.

After the FDI Client has stored the UIP executable on the FDI Client the FDI Client loads the UIP Assemblies dynamically into the memory and executes the related logic by calling the corresponding FDI specified interface functions.

Subclause 4.7 describes rules about how the FDI Client shall activate and deactivate the UIP.

4.7.2 UIP Assembly activation steps

4.7.2.1 Load

The FDI Client shall load the UIP executables by using the LoadFrom mechanism. The .NET framework provides System.Reflection.Assembly.LoadFrom for this purpose:

The LoadFrom mechanism behaves as follows.

- LoadFrom loads the Assembly addressed with the file path and also the referenced Assemblies located within same directory. The argument string assemblyFile shall contain the file name of the UIP executable. The file name of the UIP executable represents the StartElementName described in IEC 62769-4.
- If an Assembly is loaded with LoadFrom, and later an Assembly in the “load context” attempts to load the same Assembly by display name, then this load attempt fails.
- If an Assembly with the same identity is already loaded (for example, by another UIP), then LoadFrom returns the Assembly that has been loaded before, even if a different file path was specified. Even a different file name does not matter. Only the identity of the Assembly is relevant.
- If an Assembly is loaded with LoadFrom, and the probing path includes an Assembly with the same identity (for example, in the Global Assembly Cache or an application directory), then this Assembly is loaded, even if a different file path was specified.

- `LoadFrom` requires the permissions `FileIOPermissionAccess.Read` and `FileIOPermissionAccess.PathDiscovery`, or `WebPermission`, on the specified path.
- `LoadFrom` loads the assembly into the default Application Domain.
- If a native Assembly image (generated by ngen.exe) exists for the specified file path, then it is not used. The Assembly cannot be loaded as domain neutral, i.e., the Assembly cannot be shared between Application Domains.

This behavior enforces deployment rules as follows.

- Rules regarding Assembly dependencies (see 4.7.2.4.2).

The FDI Client shall only use `LoadFrom`. The use of other .NET Assembly loading/object creation means is not allowed.

- Rules regarding shared Assemblies (see 4.7.2.4.3).
- A pre-compiled processor-specific machine code cannot be used.
- The security aspects regarding loading and execution of Assemblies are described in 4.9.

4.7.2.2 Create

Creating an instance of the UIP Assembly works using the .net library functions `System.Reflection.Assembly.GetTypes` and `System.Activator.CreateInstance`. The FDI type library declares a “custom attribute” named `UIPActivationClass`. This attribute shall only be added to the object implementing the interface `IDtmUiFunction` that actually implements the UIP start-up function. The attribute `UIPActivationClass` shall be used once only.

The FDI Client can now use `System.Reflection` services to clearly determine the UIP implemented activation procedure.

NOTE 1 Function System.Reflection.Assembly.GetTypes can be used to query the interface IDtmUiFunction.

NOTE 2 Function System.Attribute.GetCustomAttributes can be used for reading the additional custom attributes.

NOTE 3 The result of function invocation System.Activator.CreateInstance is an object of type IDtmUiFunction.

A data type cast is needed.

4.7.2.3 Activate

Invocation of function `IDtmUiFunction.Init` finally activates the UIP for the user.

4.7.2.4 External libraries

4.7.2.4.1 General

UIP Assemblies can depend on external libraries (3rd party libraries) and other Assemblies, for example, specific user control libraries. FDI Clients do not perform installation of UIPs, rather they dynamically load and execute the UIP. To support this usage, as well as the requirement to prevent possible problems of conflicting Assemblies, rules are specified for external libraries.

External libraries shall:

- be contained within the FDI Package;
- not require Microsoft Installer (MSI) installation;
- not require entries in the Windows Registry or the Global Assembly Cache;
- adhere to the access restrictions described in 4.9.2;

- be compatible with the platforms described in 4.1.2.

4.7.2.4.2 Loading of external libraries

The FDI Client loads the UIP Assembly, containing the UIP main class implementing interface `IDtmUiFunction`, by invocation of the .NET framework function `LoadFrom`. Referenced Assemblies that are stored in the same directory are automatically loaded together with this .NET Assembly. Referenced Assemblies that are stored in other locations (for example, in a sub-directory) have to be loaded explicitly by the UIP itself.

The UIP shall load such Assemblies also by invocation of the .NET framework function `LoadFrom`. Loading Assemblies with other .NET framework methods is not allowed.

Usage of external libraries shall not break the self-containment requirement for FDI Packages; all external libraries shall be included in the FDI UIP Package

4.7.2.4.3 Loading of shared external libraries

An external library is a shared external library if a related .NET Assembly identity can be used from different UIP executables. The identity of a .NET Assembly matters. Installation path and Assembly filename are not relevant.

Usage of shared libraries shall not break the self-containment requirement for FDI Packages. Each of the delivered FDI Packages shall be shipped with all required UIP related libraries. The sharing mechanism comes from the .NET framework implemented optimization mechanism.

If a shared Assembly is used, then the following rules apply.

- Any incompatible change to the shared Assembly shall lead to a new identity, for example, different version number.
- Shared Assemblies shall not presume to be loaded from a specific installation path, for example, rely on the fact that some files are stored in the same directory or in a sub-directory.
- Static variables in shared Assemblies are also shared if the Assembly is loaded into the same Application Domain. Thus static variables shall not have side effects in such scenarios. External shared libraries shall not declare static variables.
- Because of the self-containment rule defined for the FDI Package, shared Assemblies shall be deployed with all FDI Packages using a shared Assembly.

4.7.2.5 UIP Constructor invocation

Constructor and destructor implementation shall not throw exceptions. The constructor logic shall be limited to instantiate the object in terms of the internal data structure. The destructor logic shall be limited to destroy the object in terms of releasing memory resources. The constructor and the destructor shall not:

- Invoke any call-back to the FDI Client.
- Invoke any user interaction.

4.7.3 UIP Assembly deactivation steps

4.7.3.1 Deactivate

For UIP deactivation the FDI Client shall call the interface `IDtmUiFunction.BeginClose` and `IDtmUiFunction.EndClose`. On successful execution the UIP shall release all resources and the FDI Client shall delete all references to the UIP instance. The .NET garbage collector finally disposes the UIP runtime object.

4.7.3.2 Dispose

A .NET Assembly that is loaded into a process respectively into the related `ApplicationDomain` is never unloaded, except if the `ApplicationDomain` itself is destroyed. That means if the FDI Client loads a UIP Assembly into the default `ApplicationDomain`, then these Assemblies and all dependent Assemblies are never unloaded unless the application is closed.

The UIP Assemblies shall be developed with this .NET framework behavior in mind. To reduce the memory consumption the following rules apply.

- Minimize the use of static variables, because these increase the memory consumption of the Assembly.
- Move UIP functionality that is not always (or rarely) needed to separate Assemblies. These Assemblies are then only automatically or manually loaded when the corresponding code is executed.
- Use shared Assemblies whenever possible.
- The FDI Client can execute .NET Assemblies in a separate Application Domain in order to have the ability to unload them.

4.8 Interaction between an FDI Client and a UIP

4.8.1 Handling of standard UI elements

UIPs shall delegate the presentation and handling of standard UI elements to the FDI Client. The standard UI elements are

- UI Actions with standardized semantics (Apply/Close/Online Help), and
- UIP Specific status information.

To ensure a consistent user interface interaction across UIPs from different vendors, a UIP may delegate presentation and handling of additional UIP specific actions to the FDI Client. Nonetheless UIPs are allowed to implement non-standard UI actions within their own UI area.

The set of standard UI actions and their respective semantics is fixed. However, the availability of these actions may change at any time depending on the internal state of the UIP. The set of additional UIP specific actions and their individual availability is not fixed. A UIP may add, remove, rename, enable or disable the UIP specific actions at any time depending on its requirements. The UIP has to inform the FDI Client whenever the availability of its standard actions or UIP specific actions changes (see events `IStandardActions.StandardActionItemSetChanged` and `IApplicationSpecificActions.ApplicationSpecificActionItemSetChanged`).

An FDI Client may use dedicated UI elements, e.g. button controls, to provide direct access to the standard actions, as well as indirectly invoke them in the context of user interaction with other FDI Client UI elements. FDI Client shall always show all custom actions exposed by a UIP with dedicated UI elements.

4.8.2 Non-blocking service execution

4.8.2.1 FDI Client internal functions

The implementation of function Begin*OperationName* shall copy the content of Argument `asyncState` into member `AsyncState` of the returned `IAsyncResult` object.

The productive (time consuming) part of the function named *OperationName* shall be performed in a different thread. The synchronization with the calling thread is handled via the `AsyncWaitHandle` object (class `WaitHandle`), which is also a member of the `IAsyncResult` object.

When processing of the productive part of the function named *OperationName* has finished the `IAsyncResult` objects attribute `IsCompleted` shall be set to `True`. If the `AsyncCallBack` argument value is valid (not equal NULL) the FDI Client notifies the UIP using the callback.

The implementation of Cancel*OperationName* uses the argument `IAsyncResult` to identify the service that has been started with Begin*OperationName*. If Begin*OperationName* started an OPCUA service, the FDI Client shall call the OPCUA defined Cancel service.

4.8.2.2 UIP internal functions

The management of multiple asynchronous services in parallel shall be managed using the `AsyncState` object.

The `IAsyncResult` object returned by Begin*OperationName* contains the `WaitHandle` object. The UIP shall perform its own thread synchronization using the `WaitHandle` object.

4.8.2.3 Non-blocking service execution sequence

The following shows the interaction sequence between the FDI Client and the UIP. The thread management mechanisms implemented inside the FDI Client are not shown. The Interaction between an FDI Client and an FDI Server is based on Request/Response pattern. The FDI Client service request matches with the Begin*OperationName*. The AsyncCallback invocation matches with receiving the Client service response. End*OperationName* conveys the response contained results. Implementation of the non-blocking service execution does not require any thread management inside the FDI Client. Figure 4 shows an example of a IAsyncPattern based Asynchronous service execution.

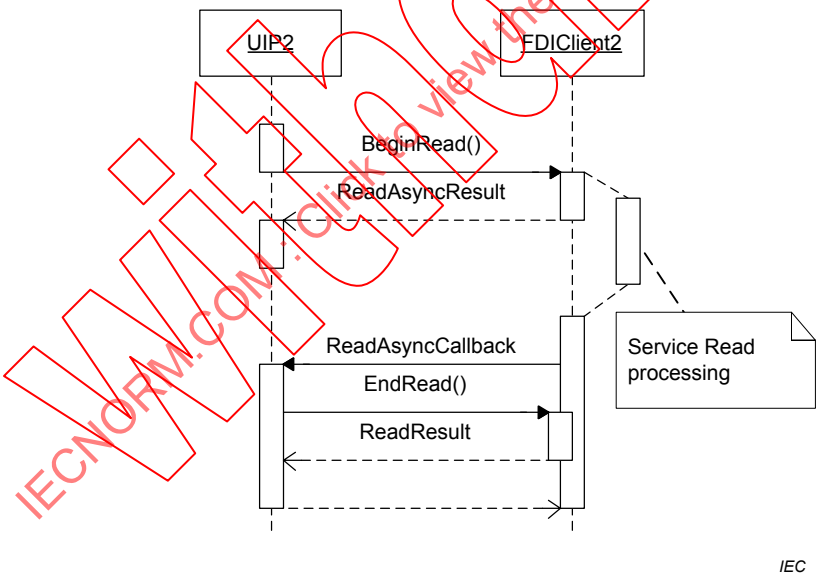


Figure 4 – IAsyncPattern based asynchronous service execution example

4.8.3 Blocking service execution

The FDI Client provided interfaces allow performing synchronous Information Model access by using the functionality described in 4.8.1 in a way shown in Figure 5.

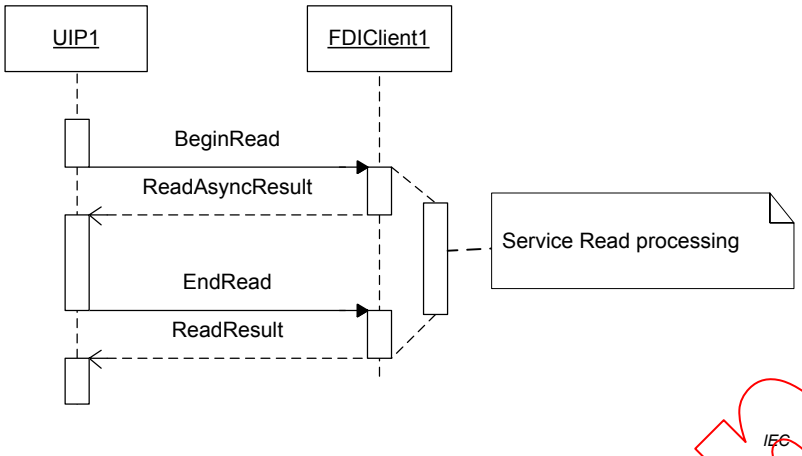


Figure 5 – Blocking service execution example using IAsyncResult based pattern

`ReadAsyncResult` is the object implementing the interface `IAsyncResult`.

4.8.4 Cancel service execution

Some services specified for the interface `IDeviceModel` (see Table 3) support canceling a started service by means of the function Cancel*IOperationName*. The following Figure 6 will illustrate the processing sequence based on the Read service example.

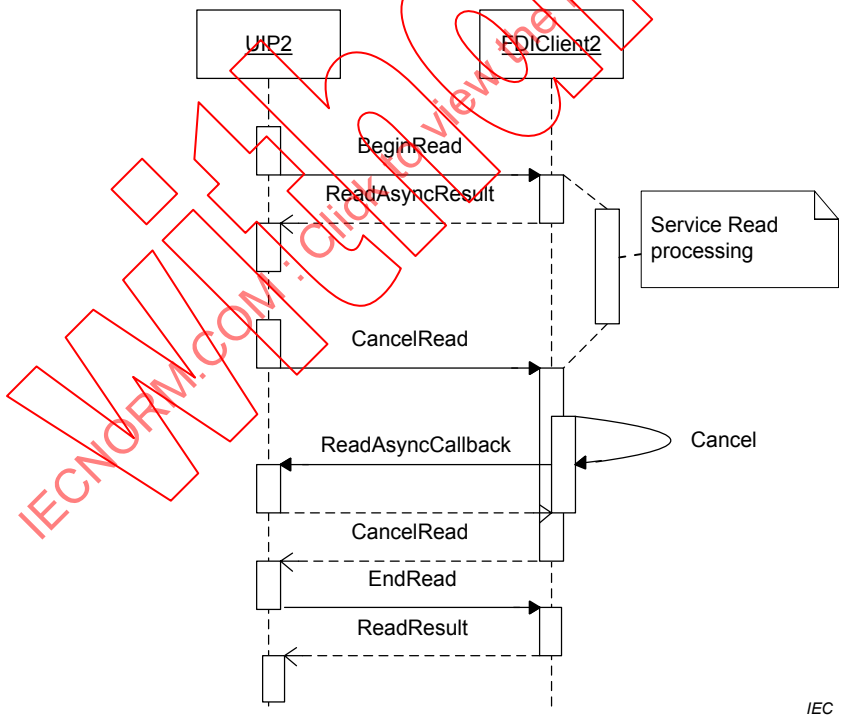


Figure 6 – Cancel service processing sequence example

The invocation of `CancelRead` triggers the FDI Client internal functions needed to cancel the active read operation. The FDI Client may not be able to cancel the operation immediately, but it should do so as soon as possible. Once the operation has been cancelled, the FDI Client notifies the UIP through the `ReadAsyncCallback`. The UIP shall then call the `EndRead` function.

NOTE A general challenge implementing this pattern is to handle race conditions properly on both sides (UIP2 and FDIClient2). If the FDI Client has forwarded the service execution via an OPC UA service, the actual service execution will run inside the FDI Server.

Depending on how the UIP is hosted there may be three independently working processes. Therefore the cancel request (sent by the UIP) may appear right after the FDI Server has already finished the service request. The related response sent by the FDI Server may have arrived at the FDI Client (or not). The FDI Client may invoke the ReadAsyncCallBack while the UIP invokes the CancelRead.

`ReadAsyncResult` is the object implementing the interface `IAsyncResult`.

4.8.5 Threading

4.8.5.1 Implementation rules

The UIP shall be able to receive calls in any thread.

The UIP shall not block the calls coming from the FDI Client.

The UIP shall not use the FDI Client thread to signal back the callback to the FDI Client itself. This is to prevent deadlocks and endless loops.

The UIP shall not run synchronous operations as described in 4.8.3 in the user interface thread: The user interface thread of a process shall be dedicated to receive user inputs and perform drawing tasks only.

The UIP and FDI Client shall not block the user interface thread. The user interface shall always stay responsive. The user interface thread is shared between the different FDI user interface related objects for user input and drawing operations. If one object blocks this thread in order to perform some processing, this would affect the responsiveness of other user interfaces.

The UIP and FDI Client shall not block a Begin*OperationName* method call: A Begin*OperationName* method shall only start an asynchronous operation. The caller shall not be blocked.

4.8.6 Timeout

The interfaces referred in Clause 5 enable asynchronous service execution. The time for the execution of such services depends on performance constraints related to: bus communication, FDI Client/FDI Server performance. The rules listed below target the system interoperability regarding the prevention of “Race Conditions”. The general rule is that the component is allowed to manage timeout handling only for those processes that are completely under the control of that component. The following list shows which elements of the entire system are allowed to implement the timeout detection function.

- UIP: The UIP shall not implement timeout detection.
- Business Logic: The Business Logic shall not implement timeout detection (FDI Package).
- FDI Client: The FDI Client shall implement timeout detection. In case of OPC UA, the related support is built into the OPC UA communication stacks. Timeout detected during operations performed on behalf of the UIP shall be forwarded as negative function result codes.
- FDI Server: The FDI Server shall implement timeout detection. In case of OPC UA, the related support is built into the OPC UA communication stacks.
- Communication Server: The Communication Server implements timeout detection for the OPC UA connection according to the OPC UA Specification. Related support is built into the OPC UA communication stacks. Additionally the Communication Server implements

timeout detection limited to the network directly connected to the physical port connected to the Communication Server.

4.8.7 Exception handling

An important specification goal is to make a clear distinction between software quality problems and anticipated processing states. Therefore the specification defines two general Exception categories:

a) Exceptions that indicate software states or events that have not been anticipated during the software development are considered as software quality issues (Run-time error).

b) Exceptions indicating anticipated software operation failures.

Examples of software quality issues indicated by exceptions are:

- function argument type mismatch;
- function argument value range mismatch;
- division by zero;
- NULL Pointer reference.

Examples of anticipated error handling are:

- communication problem handling;
- general IO data processing;
- user input errors.

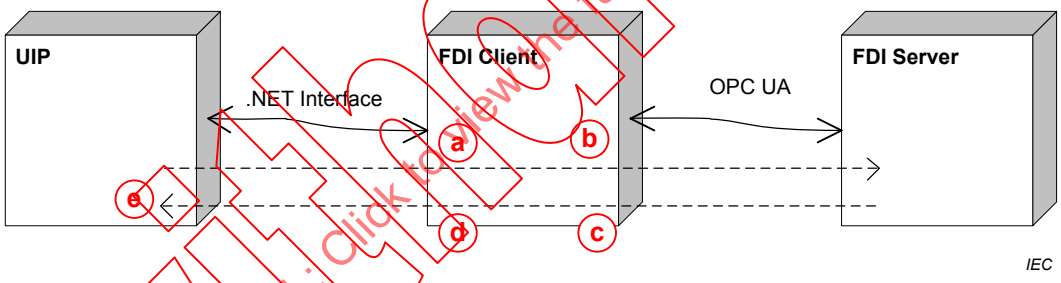


Figure 7 – Exception source

According to the FDI Architecture exceptions can occur in different steps of the service processing, see Figure 7:

a) passing the request from the UIP to the FDI Client:
b) request forwarding inside the FDI Client;
c) processing the response from the FDI Server;
d) forwarding the response to the UIP;
e) response processing inside the UIP.

Service processing problems detected inside the FDI Server and beyond are handled through OPC UA defined service results.

Regarding the implementation of the `IAsyncResult` pattern the following rules apply.

– Any failure occurring with step a) shall be reported by an exception thrown by the Begin*OperationName*.
– Any failure occurring during steps b) to e) shall be handled by the corresponding component. The execution of the End*OperationName* shall then report the failure via an exception.

4.8.8 Type safe interfaces

The Information Model hosts device variables of different types. The values of such variables are transferred using the class DataValue (FDI Interfaces and Data Types.CHM).

The Device Access Services support writing or reading multiple variables within one service. The data type chosen for data transport is `DataValue` implementing the type safe transport because of the `DataValue` property `Datatype` describing the value data type by means of `Datatype` enumeration. Because the `DataValue` property Value get/set functions use data type Object to convey the actual value the data receiver (UIP) shall verify the data type before data processing.

4.8.9 Globalization and localization

The default locale for UIP is English/(US).

Optional language support is allowed according to market needs.

UIP localization support can be implemented through resource files (.res(x)) or satellite Assemblies.

The FDI Client shall set the locale and country information that shall be used by the UIP by means of the arguments `currentRegion` and `currentCulture` that are submitted with the invocation of method `Fdi.Dtm.Ui.IDtmUiFunction.Init`. The UIP shall not derive locale information via the `Thread.CurrentUICulture`. The data type for `currentRegion` is `RegionInfo` defined in the .NET namespace `System.Globalization`. The data type for `currentCulture` is `CultureInfo` defined in the .NET namespace `System.Globalization`.

4.8.10 WPF Control handling

If a UIP implementation is based on WPF `UserControl` the UIP inherits the interface from the class `UserControl`, which means there will be more methods attributes and events available for the FDI Client that are not covered by the FDI specification. Vice versa a UIP implements the accessibility function. The related rules on the one hand touch the quality of the UIP product and on the other hand the interoperability.

4.8.11 Win Form handling

If a UIP implementation is based on Windows Forms the UIP inherits from the class `UserControl`, which means there will be more methods attributes and events available for the FDI Client that are not covered by the FDI specification. Vice versa a UIP implements the accessibility function. The related rules on the one hand touch the quality of the UIP product and on the other hand the interoperability. The FDI Client shall make thread-safe calls to the `Windows.Forms` controls.

4.9 Security

4.9.1 General

The goal of security is to protect a system against threats compromising the system stability, integrity and sensitive data.

System wide security begins with the design process, which is out of the standardization scope. From the system perspective security is about controlling access to resources, such as application components, data, and hardware. The .NET framework provides support for constraining access to resources. The system security is based on control over access permissions.

A different approach is based on certification and authentication. Since any FDI Package needs compliance testing and certification the presumption is that such certified FDI Packages don't pose any threats to a system. This means a UIP could be executed with full trusted permissions.

While an over-constrained system could lead into functional problems an un-constrained permissions can be seen as security threat. Thus Subclause 4.9 represents a compromise between both ways.

4.9.2 Access permissions

4.9.2.1 General

The access permissions for a UIP are enforced by the .net CLR run-time system following the security policy. The security policy is a configurable set of rules defining the constraints for resource access. Only administrators shall modify or customize the security policy according to the specific needs of their organizations. The CLR runtime grants permissions to both Assemblies and Application Domains based on the security policy.

Identity permissions represent characteristics that identify an Assembly. The CLR grants identity permissions to an Assembly based on the information it obtains about the Assembly.

4.9.2.2 UIP permissions

The UIP access permissions defined in this part of IEC 62769 are specified according to use cases that need to be implemented with UIP as follows.

a) Reference data bases and help files (UIP Supplementary data) shall be provided with UIP and stored within in the UIP installation folder on the FDI Client. The access permission to this folder is read-only.

b) UIP specific data like Valve Signatures shall be stored in the Information Model and accessed by means of the EDD element.

c) The export/import use case shall be supported by allowing the user to save and load data from a user specified folder. The access permissions to this folder are defined by means operating system administrated user credentials. The export/import function enables data migration, backup/restore.

d) User settings, preferences or data caching shall be done via the services SaveUserSettings and LoadUserSettings specified in IEC 62769-2.

e) Launching of an Active-X component is not allowed.

f) Sharing of UIP specific data can be done either through the Information Model or by means of the export import function described in c).

g) If a printer is available, access to that printer is allowed.

h) A UIP executable shall not implement role based access permission constraints.

i) A UIP shall not access the operating system registry.

The FDI Client shall grant the following list of minimum set of permissions:

a) A UIP executable is allowed to read UIP supplementary data files from the installation directory and below (see 4.2). The UIP is not allowed to browse the file system above its installation root.

b) A UIP executable shall not perform internet access.

c) A UIP executable shall not perform local area network (LAN) access.

4.9.2.3 Implementation rules

Sandboxing enables running the code in an environment with restricted permissions. An FDI Client shall limit the access permissions given to a UIP.

The FDI Client can run the UIP within an Application Domain providing a sandbox for the UIP. The Application Domain is used for running the partially trusted UIP with permissions that define the availability of protected resources when running within that Application Domain. The UIP that runs inside the Application Domain is bound by the permissions associated with the Application Domain and is allowed to access only the specified resources.

The FDI Client shall use the function `System.AppDomain.CreateDomain(String, Evidence, AppDomainSetup, PermissionSet, StrongName[])` method overload to specify the permission set for the UIP that runs in a sandbox. This overload enables the FDI Client to specify the exact level of code access security specified in 4.9.2.2.

NOTE Assemblies that are loaded into an Application Domain by using this overload can either have the specified grant set only, or can be fully trusted. The Assembly is granted full trust if it is in the Global Assembly Cache or listed in the fullTrustAssemblies (the StrongName) array parameter.

Only Assemblies known to be fully trusted should be added to the fullTrustAssemblies list. The list of trusted assemblies is managed by the operating system.

The PermissionSet assigned to the Application Domain running the UIP (UIP-Sandbox) shall be initialized with

PermissionSet(PermissionState.None)

The UIP permission set shall contain:

a) `FileDialogPermissionAccess`
b) `FileIOPermissionAccess`
c) `UIPermissionWindow`
d) `SecurityPermissionFlag.Execution`
e) `ReflectionPermission`

4.9.3 Code identity concept

The ability to uniquely identify UIP executables contributes to the system security.

As earlier described in 4.3 UIP executables shall be signed with strong names. Strong names signed .NET Assemblies enable:

– Unique identification of UIP executables.
– Code integrity verification.

NOTE The benefit of strong named UIP executable is lost if this Assembly dynamically loads other library Assemblies that are not signed with strong names.

5 Interface definition

The following tables specify the mapping between the abstract services specified in IEC 62769-2 and the corresponding .NET implementation which is found in the type library file “FDI.DLL” and a related help file “FDI Interfaces and Data Types.chm”.

NOTE The Files “FDI.DLL” and “FDI Interfaces and Data Types.chm” can be obtained from the fieldbus organizations. (See also http://www.fdi-cooperation.com).

Table 2 specifies the mapping of the Base Property Services.

Table 2 – Base Property Services

Abstract Service	.NET Implementation
GetDeviceAccessInterfaceVersion	IDeviceAccess.Version
GetOnlineAccessAvailability	IDeviceAccess.OnlineAccessAvailable

Table 3 specifies the mapping of the Device Model Services.

Table 3 – Device Model Services

Abstract Service	.NET Implementation
Browse	IDeviceModel.BeginBrowse IDeviceModel.CancelBrowse IDeviceModel.EndBrowse
Read	IDeviceModel.BeginRead IDeviceModel.CancelRead IDeviceModel.EndRead
Write	IDeviceModel.BeginWrite IDeviceModel.CancelWrite IDeviceModel.EndWrite
CreateSubscription	IDeviceModel.BeginCreateSubscription IDeviceModel.EndCreateSubscription
Subscribe	IDeviceModel.BeginSubscribe IDeviceModel.EndSubscribe
Unsubscribe	IDeviceModel.BeginUnsubscribe IDeviceModel.EndUnsubscribe
DeleteSubscription	IDeviceModel.BeginDeleteSubscription IDeviceModel.EndDeleteSubscription
DataChangeCallback	DataChangeCallback

Table 4 specifies the mapping of the Access Control Services.

Table 4 – Access Control Services

Abstract Service	.NET Implementation
InitLock	IAccessControl.BeginInitLock IAccessControl.EndInitLock
ExitLock	IAccessControl.BeginExitLock IAccessControl.EndExitLock

Table 5 specifies the mapping of the Direct Access Services.

Table 5 – Direct Access Services

Abstract Service	.NET Implementation
InitDirectAccess	IDirectAccess.BeginInitDirectAccess IDirectAccess.EndInitDirectAccess
ExitDirectAccess	IDirectAccess.BeginExitDirectAccess IDirectAccess.EndExitDirectAccess
Transfer	IDirectAccess.BeginTransfer IDirectAccess.EndTransfer

Table 6 specifies the mapping of the Hosting Services.

Table 6 – Hosting Services

Abstract Service	.NET Implementation
GetClientTechnologyVersion	Fdi.Frame.IFrame.Version
OpenUserInterface[a)]	Fdi.Frame.Ui.IFrameUi.BeginOpenDtmUiModal Fdi.Frame.Ui.IFrameUi.EndOpenDtmUiModal
CloseUserInterface[a)]	Fdi.Dtm.Ui.CloseMeRequestHandler [c)]
LogAuditTrailMessage	Fdi.Frame.IAuditTrail.Notify
SaveUserSettings	Fdi.Frame.IUserSettings.SaveUserSettings
LoadUserSettings	Fdi.Frame.IUserSettings.LoadUserSettings
Trace	Fdi.Frame.ITrace.Trace
ShowMessageBox	Fdi.Frame.Ui.IFrameUi.ShowMessageBox
ShowProgressBar	Fdi.Frame.Ui.IFrameUi.ShowProgress
CancelCallback	Fdi.Frame.Ui.CancelEventHandler
UpdateShowProgressBar	Fdi.Frame.Ui.IProgressUi.UpdateProgress
EndShowProgressBar	Fdi.Frame.Ui.IProgressUi.EndProgress
DefaultResult	System.Windows.MessageBoxResult
ButtonSet	System.Windows.MessageBoxButton
AcknStyle	System.Windows.MessageBoxImage

a) Functions `OpenUserInterface`, `CloseUserInterface`, `OpenModalUserInterface` shall only be started using the operation pattern described in 4.8.2.3.

b) Functions are to be used to manage an additional UIP.

c) To be used by the UIP to close itself.

Table 7 specifies the mapping of the UIP Services.

Table 7 – UIP Services

Abstract Service	.NET Implementation
Activate	Fdi.Dtm.Ui.IDtmUiFunction.Init
Deactivate	Fdi.Dtm.Ui.IDtmUiFunction.BeginClose [a)] Fdi.Dtm.Ui.IDtmUiFunction.EndClose [a)]
SetSystemLabel	Fdi.Dtm.Ui.IDtmUiFunction.SystemGuiLabel
SetTraceLevel	Fdi.Dtm.Ui.IDtmUiFunction.TraceLevel
TraceLevel	Fdi.Frame.TraceEventType
InvokeStandardAction(*1)	Fdi.Dtm.Ui.IStandardActions.InvokeStandardAction
InvokeApplicationSpecificAction	Fdi.Dtm.Ui. IApplicationSpecificActions.InvokeApplicationSpecificAction
GetStandardActionItems	Fdi.Dtm.Ui.IStandardActions.ActionItemSet
GetApplicationSpecificActionItems	Fdi.Dtm.Ui. IApplicationSpecificActions.ApplicationSpecificActionItemSet
StandardActionItemsChangeCallback	Fdi.Dtm.Ui.IStandardActions.StandardActionItemSetChanged
ApplicationSpecificActionItemsChangeCallback	Fdi.Dtm.Ui. IApplicationSpecificActions.ApplicationSpecificActionItemSetChanged

a) The Deactivate service specified response `deactivateCancelled` maps to the exception `FdiCannotCloseUiException` to be thrown by the UIP if the UIP has problems with the deactivation.

Table 8 specifies the mapping of the base data types.

Table 8 – Base Data Types

Base data type	.NET Implementation	
Boolean	Fdi.DataTypes.BooleanValue	enum DataType.Boolean
String	Fdi.DataTypes.StringValue	enum DataType.String
ByteString	Fdi.DataTypes.BinaryValue	enum DataType.Binary
UtcTime	Fdi.DataTypes.DateTimeValue	enum DataType.DateTime
Int8	Fdi.DataTypes.SByteValue	enum DataType.SByte
Int16	Fdi.DataTypes.ShortValue	enum DataType.Short
Int32	Fdi.DataTypes.IntValue	enum DataType.Int
Int64	Fdi.DataTypes.LongValue	enum DataType.Long
Byte	Fdi.DataTypes.ByteValue	enum DataType.Byte
UInt16	Fdi.DataTypes.UShortValue	enum DataType.UShort
UInt32	Fdi.DataTypes.UIntValue	enum DataType.UInt
UInt64	Fdi.DataTypes.ULongValue	enum DataType.ULong
Float	Fdi.DataTypes.FloatValue	enum DataType.Float
Double	Fdi.DataTypes.DoubleValue	enum DataType.Double
Duration	Fdi.DataTypes.TimeSpanValue	enum DataType.TimeSpan

Table 9 specifies the mapping of the special data types.

Table 9 – Special Types

Special Data type	.NET Implementation	
Attribute Ids	Fdi.DeviceAccess.AttributeType	
Variant	Fdi.DeviceAccess.DataValue	
NodeSpecifier	Fdi.DeviceAccess.NodeSpecifier	
Data Value	Fdi.DeviceAccess.ReadResult	
Localized Text	Fdi. DataTypes.LocalizedTextValue	enum DataType.LocalizedText
Range	Fdi. DataTypes.RangeValue	enum DataType.Range
EU Information	Fdi. DataTypes.EUInfoValue	enum DataType.EngineeringUnit
Enum Value	Fdi. DataTypes.EnumValue	enum DataType.Enumerator
InnerErrorInfo	Fdi.DeviceAccess.InnerErrorInfo	
NumericRange	Fdi.DeviceAccess.ArrayIndexRange	

Data arrays can be conveyed using class `Fdi.DataTypes.ArrayValue`.

Detailed interface definition and interface documentation are available in:

– FDI.DLL (.NET Assembly)
– FDI Interfaces and Data Types.CHM (Help File)

Bibliography

FDI-2021, *FDI Project Technical Specification – Part 1: Overview*
<available at www.fdi-cooperation.com>

FDI-2022, *FDI Project Technical Specification – Part 2: FDI Client*
<available at www.fdi-cooperation.com>

FDI-2023, *FDI Project Technical Specification – Part 3: FDI Server*
<available at www.fdi-cooperation.com>

FDI-2024, *FDI Project Technical Specification – Part 4: FDI Packages*
<available at www.fdi-cooperation.com>

FDI-2025, *FDI Project Technical Specification – Part 5: FDI Information Model*
<available at www.fdi-cooperation.com>

FDI-2026, *FDI Project Technical Specification – Part 6: FDI Technology Mapping*
<available at www.fdi-cooperation.com>

FDI-2027, *FDI Project Technical Specification – Part 7: FDI Communication Devices*
<available at www.fdi-cooperation.com>

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTEGRATION DES APPAREILS DE TERRAIN (FDI) –

Partie 6: Mapping de technologies FDI

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.

La Norme internationale IEC 62769-6 a été établie par le sous-comité 65E: Les appareils et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels.

Le texte de cette norme est issu des documents suivants:

CDV	Rapport de vote
65E/349/CDV	65E/426/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 62769, publiées sous le titre général *Intégration des appareils de terrain (FDI)*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

La Commission Electrotechnique Internationale (IEC) attire l'attention sur le fait qu'il est déclaré que la conformité avec les dispositions du présent document peut impliquer l'utilisation de brevets intéressant:

a) la méthode de fourniture et d'installation des fonctionnalités spécifiques aux appareils (cf. famille de brevets DE10357276);

b) la méthode et l'appareil utilisés pour l'accès à un module fonctionnel du système d'automation (cf. famille de brevets EP2182418);

c) les méthodes et les appareils utilisés pour diminuer les exigences mémoire relatives aux applications logicielles du système de commande de processus (cf. famille de brevets US2013232186);

d) modèle d'objet d'appareil extensible (cf. famille de brevets US12/893,680).

L'IEC ne prend pas position quant à la preuve, à la validité et à la portée de ces droits de propriété.

Le détenteur de ces droits de propriété a donné l'assurance à l'IEC qu'il consent à négocier des licences avec des demandeurs du monde entier, soit sans frais soit à des termes et conditions raisonnables et non discriminatoires. A ce propos, la déclaration du détenteur des droits de propriété est enregistrée à l'IEC. Des informations peuvent être demandées à:

a) ABB Research Ltd
Claes Rytoft
Affolterstrasse 4
Zurich, 8050
Suisse

b) Phoenix Contact GmbH & Co KG
Intellectual Property, Licenses & Standards
Flachsmarktstrasse 8, 32825 Blomberg
Allemagne

c) Fisher Controls International LLC
John Dilger, Emerson Process Management LLLP
301 S. 1st Avenue, Marshaltown, Iowa 50158
Etats-Unis

d) Rockwell Automation Technologies, Inc.
1 Allen-Bradley Drive
Mayfield Heights, Ohio 44124
Etats-Unis

L'attention est d'autre part attirée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété autres que ceux qui ont été mentionnés ci-dessus. L'IEC ne saurait être tenue pour responsable de l'identification de ces droits de propriété en tout ou partie.

L'ISO (www.iso.org/patents) et l'IEC (http://patents.iec.ch) tiennent à jour des bases de données en ligne sur les brevets relatifs à leurs normes. Les utilisateurs sont encouragés à consulter ces bases de données pour obtenir l'information la plus récente concernant les brevets.

INTEGRATION DES APPAREILS DE TERRAIN (FDI) –

Partie 6: Mapping de technologies FDI

1 Domaine d'application

La présente partie de l'IEC 62769 spécifie le mapping de technologies pour les concepts décrits dans la norme d'intégration des appareils de terrain (FDI). Le mapping de technologies se concentre sur la mise en œuvre relative aux composants: Client FDI et Plugiciel d'Interface Utilisateur (UIP) qui ne sont spécifiques qu'à la plate-forme de station de travail telle que définie dans l'IEC 62769-4:2015, Annexe E.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 62541 (toutes les parties), *Architecture unifiée OPC*

IEC 61804 (toutes les parties), *Blocs fonctionnels (FB) pour les procédés industriels*

IEC 62769-1, *Intégration des appareils de terrain (FDI) – Partie 1: Vue d'ensemble*

NOTE L'IEC 62769-1 est techniquement identique à la FDI-2021.

IEC 62769-2, *Intégration des appareils de terrain (FDI) – Partie 2: Client FDI*

NOTE 1 L'IEC 62769-2 est techniquement identique à la FDI-2022.

NOTE 2 L'IEC 62769-2 est techniquement identique à la FDI-2023.

IEC 62769-4:2015, *Intégration des appareils de terrain (FDI) – Partie 4: Paquetages FDI*

NOTE L'IEC 62769-4 est techniquement identique à la FDI-2024.

IEC 62769-5, *Intégration des appareils de terrain (FDI) – Partie 5: Modèle d'Information FDI*

NOTE 1 L'IEC 62769-5 est techniquement identique à la FDI-2025.

NOTE 2 L'IEC 62769-5 est techniquement identique à la FDI-2027.

ISO/IEC 19505-1, *Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure* (disponible en anglais seulement)

ISO/IEC 29500 (toutes les parties), *Information technology – Document description and processing languages – Office Open XML File Formats* (disponible en anglais seulement)

3 Termes, définitions, abréviations, acronymes et conventions

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC 62769-1, ainsi que les suivants s'appliquent.

3.1.1
domaine d'application
environnement isolé où s'exécutent les applications

3.1.2
assemblage
informations de version réutilisables fournissant et auto décrivant le bloc de construction d'une application CLR

Note 1 à l'article: L'abréviation "CLR" est dérivée du terme anglais développé correspondant "Common Language Run-time".

3.1.3
bibliothèque de types FDI
assemblage contenant les interfaces et les types de données qui sont utilisés pour l'échange de données et l'interaction entre un UIP et un Client FDI

Note 1 à l'article: L'abréviation "UIP" est dérivée du terme anglais développé correspondant "User Interface Plug-in".

Note 2 à l'article: L'abréviation "FDI" est dérivée du terme anglais développé correspondant "Field Device Integration".

3.1.4
cache d'assemblages global
cache de codes à l'échelle de la machine qui stocke des assemblages spécifiquement désignés pour être partagés par plusieurs applications

3.1.5
registre Windows
base de données définie par le système dans laquelle les applications et les composants système stockent et récupèrent les données de configuration

3.2 Abréviations et acronymes

Pour les besoins du présent document, les abréviations et acronymes de l'IEC 62769-1, ainsi que les suivants s'appliquent.

CLR	Common Language Run-time (moteur d'exécution du langage commun)
MSI	Microsoft Installer (programme d'installation Microsoft)
WPF	Windows Presentation Foundation (fondation de présentation Windows)
UML	Unified Modeling Language (langage de modélisation unifié)

3.3 Symboles

Les figures du présent document utilisent les symboles graphiques selon l'ISO/IEC 19505 (UML 2.0).

4 Concepts techniques

4.1 Généralités

4.1.1 Vue d'ensemble

En 4.1.2, 4.2, 4.3, 4.4, et 4.5, le présent document décrit d'abord la base de la technologie pour la mise en œuvre de l'UIP, l'environnement matériel et logiciel, y compris les règles de mise en œuvre connexes. L'Article 4 suit une approche orientée cycle de vie (cas d'utilisation).

Le Paragraphe 4.6 décrit les procédures de déploiement des copies et les règles de mise en œuvre connexes pour l'UIP et le Client FDI.

L'instanciation et la terminaison de l'exécutable de l'UIP sont décrites en 4.7.

Le Paragraphe 4.8 définit les règles d'interaction entre un Client FDI et l'UIP.

Les définitions relatives à la sécurité sont données en 4.9.

Les définitions d'interface de service pour le Client FDI et l'UIP se trouvent à l'Article 5.

4.1.2 Plates-formes

L'UIP et le Client FDI doivent être construits sur le .NET Framework de Microsoft et exécutés dans le Common Language Run-time .NET.

L'ensemble minimal des appareils d'E/S pris en charge par la station de travail est: une souris, un clavier, un écran couleur de résolution 1024 x 768 pixels.

Le Tableau 1 suivant énumère toutes les technologies et leurs éditions conformes aux composants FDI.

Tableau 1 – Référence de l'édition de technologie

Technologie	Norme	Edition
.NET	N/A	CLR4 pour la mise en œuvre de l'UIP
EDDL	IEC 61804	2014
OPC UA (Parties 1-8)	IEC 62541	2015 (à paraître)
Convention de Paquetage Ouvert (OPC)	ISO/IEC 29500	2011
Langage de balisage extensible (XML)	N/A	W3C, 1.0 (cinquième édition)

4.1.3 Bibliothèque de types FDI

Les Services d'Accès à l'Appareil et les Services d'UIP peuvent être modélisés comme des interfaces .NET conformes aux arguments de types de données .NET. Ces interfaces et ces types de données sont utilisés pour l'échange de données et l'interaction entre l'UIP et le Client FDI. Pour l'objet relatif au traitement des erreurs d'exécution durant les appels de méthodes d'interfaces, les classes d'exceptions .NET sont définies.

Les interfaces, types de données et classes d'exception FDI .NET sont définis dans une bibliothèque de types FDI unique. La bibliothèque de types FDI est un assemblage à nom fort.

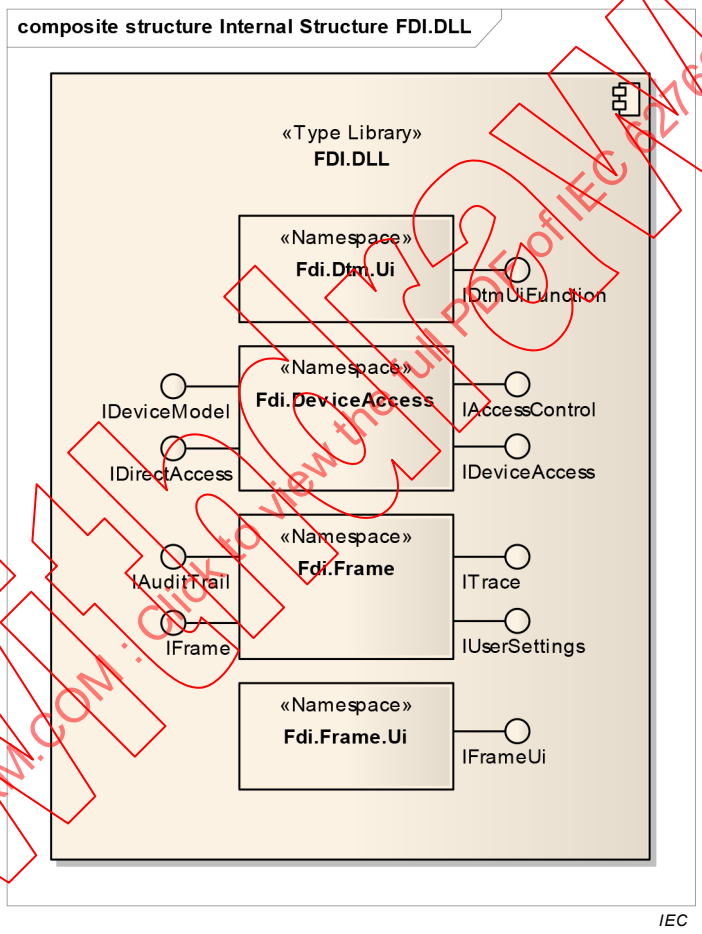
La bibliothèque de types FDI est signée avec une clé unique. La bibliothèque de types FDI doit être installée dans le cadre de l'installation du Client FDI et non avec un UIP.

Les bibliothèques de types FDI ne doivent pas être enregistrées dans le Cache d'assemblages global.

Le Client FDI doit installer les Versions de Bibliothèque FDI pour l'ensemble des Versions de technologie qu'il prend en charge.

La bibliothèque de types FDI doit être installée de manière à ce qu'elle soit partagée entre l'UIP et le Client FDI.

La Figure 1 montre la structure de bibliothèque de types FDI.



Anglais	Français
Composite structure internal structure FDI.DLL	Structure composites/Structure interne FDI.DLL
Type Library	Bibliothèque de types
Namespace	Espace de noms

Figure 1 – Structure de la bibliothèque de types FDI

NOTE Le diagramme de structure composite ne montre que les interfaces de base qui mettent en œuvre les interfaces définies dans l'IEC 62769-2.

4.2 Représentation de l'UIP

La variante de l'UIP peut contenir un seul ou plusieurs modules exécutables (Assemblage .NET) et leurs fichiers supplémentaires connexes (par exemple: fichiers

ressources). Le module exécutable de la variante de l'UIP est appelé "exécutable de l'UIP". Le(s) fichier(s) supplémentaire(s) de la variante de l'UIP est/sont appelé(s) "supplément(s) de l'UIP".

Le(s) supplément(s) de l'UIP est/sont stocké(s) dans un ou plusieurs sous-dossiers du répertoire d'installation de l'exécutable de l'UIP

EXEMPLE Des exemples de fichiers de données supplémentaires de l'UIP comprennent les fichiers de ressources et les données de configuration d'application.

Le RuntimeId d'une variante de l'UIP doit être ".NET Framework CLR4", voir l'IEC 62769-4.

La variante de l'UIP doit être autonome. Toutes les bibliothèques nécessaires de l'UIP (assemblages .NET) exigées par une variante de l'UIP sont stockées dans le même dossier.

4.3 Représentation de l'exécutable de l'UIP

La mise en œuvre de l'UIP dépend du type d'éléments d'interface utilisateur qui peuvent être intégrés dans l'environnement d'hébergement de l'interface utilisateur du Client FDI. L'UIP doit être mis en œuvre comme suit: NET `System.Windows.Forms` classe `UserControl` ou Windows Presentation Foundation (WPF) `System.Windows.Controls` classe `UserControl`.

Les exécutables de l'UIP et leurs bibliothèques nécessaires doivent avoir des noms forts. La signature d'un assemblage ayant un nom fort peut être effectuée en utilisant une clé autogénérée.

NOTE L'identité d'assemblages de noms forts se compose d'un nom, d'une version, d'une culture, d'un jeton de clé publique et d'une signature numérique.

Les exécutables de l'UIP et leurs bibliothèques nécessaires doivent être livrés avec le fichier contenant la clé publique afin de permettre la vérification de l'assemblage.

4.4 Règles de compatibilité de l'exécutable de l'UIP

Les informations relatives à la version fournie par les composants UIP comprennent:

<Major>.<Minor>.<Build Number>.<Revision> (c'est-à-dire: <Majeure>.<Mineure>.<Numéro de build>.<Révision>)

Les composants de l'UIP utilisant la même identité (UipId/IEC 62769-5) qui montrent une valeur différente en position <Major> (c'est-à-dire <majeure>) ne sont pas compatibles les uns avec les autres. Toute autre différence montrée dans l'information relative à la version entre les mêmes identités de composants UIP signifie que les identités de ces composants de l'UIP sont compatibles. Un composant de l'UIP plus récent peut écraser un composant de l'UIP plus ancien sans interrompre la fonctionnalité prévue.

La plate-forme cible de compilation pour l'UIP doit être "anyCPU". Si ce n'est pas possible, l'UIP doit être livré en deux variantes. Une variante de l'UIP doit être compilée pour la plate-forme cible "x86". La deuxième variante de l'UIP doit être compilée pour la plate-forme cible "x64". La plate-forme cible de compilation doit être décrite dans le fichier catalog.xml qui est défini dans l'IEC 62769-4. Ce fichier catalog.xml contient un élément xml "CpuInformation" qui décrit la variante du Plugiciel d'Interface Utilisateur. Les valeurs permises qui doivent être utilisées dans l'élément xml "CpuInformation" sont "anyCPU", "x86" ou "x64".

4.5 Versions permises du CLR (Common Language Run-time) .NET

4.5.1 Généralités

Des versions CLR (Common Language Run-time) spécifiques sont publiées pour l'exécution de composants logiciels intégrés avec des versions .NET Framework spécifiques. La version

4.0 de .NET CLR est utilisée pour exécuter des composants logiciels intégrés avec .NET Framework 4.0. Les composants .NET ne sont construits que pour une seule version CLR, mais peuvent être capables de fonctionner aussi sous une version plus récente de CLR.

Les Clients FDI peuvent être construits sur la base du CLR version 4.0 ou de versions ultérieures. Un Client FDI doit se rendre compte des conditions suivantes lors du démarrage d'un UIP.

- Lorsque l'UIP à démarrer a été construit pour la même exécution, l'UIP peut être démarré dans le Client FDI comme d'habitude.
- Lorsque l'UIP à démarrer a été construit avec une autre version CLR et n'est pas compilé pour la version exécutable actuelle de CLR, le Client FDI doit démarrer l'UIP dans un processus de substitution avec une version CLR appropriée. Plus de détails sont donnés en 4.5.2.

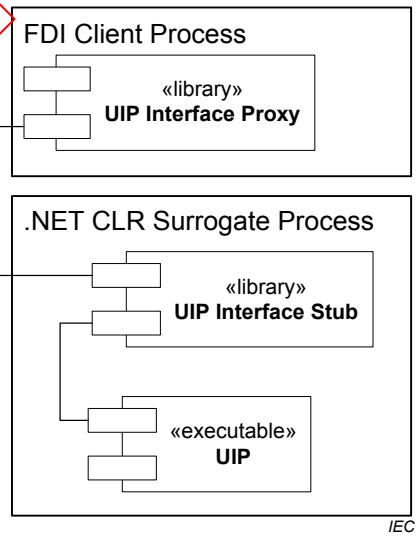
Prenant ce comportement en compte, un UIP doit être développé pour la version CLR 4.0 ou une version ultérieure. Dans le cas où les versions CLR ne seraient pas compatibles, l'UIP doit être démarré dans un processus séparé. L'UIP ne sera alors pas affiché comme un module intégré au sein du Client FDI. Il incombe au Client FDI de réaliser le processus de substitution.

4.5.2 Stratégie de compatibilité CLR

Dans le futur, il sera permis de construire les Clients FDI et UIP sur différentes versions incompatibles du CLR.

Si un Client FDI détecte qu'un UIP nécessite un CLR qui n'est pas compatible avec le Client FDI, le Client FDI peut utiliser une classe "proxy" (serveur mandataire) qui permet une interaction avec l'UIP construit, utilisant une version différente du CLR.

Le Client FDI charge un exécutable de l'UIP du serveur mandataire, crée une instance de la classe "proxy" et délègue l'exécution de l'UIP à ce serveur mandataire. Le serveur mandataire démarre un processus avec le CLR nécessaire et exécute l'UIP dans ce processus de substitution. Les classes "proxy" fournissent les interfaces FDI normalisées. Le Client FDI peut utiliser ces interfaces pour interagir avec l'UIP exécuté dans le processus de substitution.



Anglais	Français
FDI Client Process	Processus Client FDI
"Library"	"Bibliothèque"

Anglais	Français
UIP Interface Proxy	Proxy de l'interface de l'UIP
.NET CLR Surrogate Process	Processus de substitution CLR .NET
"Library" UIP Interface Stub	"Bibliothèque" Elément de remplacement de l'interface de l'UIP
"executable" UIP	"exécutable" UIP

Figure 2 – Processus de substitution .NET

4.5.3 Comment identifier une plate-forme cible .NET d'un UIP

L'information relative à la version CLR de la plate-forme de la cible .NET pour laquelle un certain assemblage est compilé peut être extraite au moyen des fonctions de la bibliothèque .NET Framework (voir Figure 3).

```
clrVersion = Assembly.LoadFrom(<Assembly Path>).ImageRuntimeVersion;
```

IEC

Anglais	Français
Assembly Path	Trajet de l'assemblage

Figure 3 – Identification de la version exécutable

NOTE L'environnement de développement Intégré (IDE) Visual Studio[1] 2008 et 2010 permet aux développeurs de sélectionner la cible .NET Framework. La sélection d'une cible .NET Framework plus ancienne que la base relative à la version actuelle de Visual Studio IDE crée automatiquement un fichier de configuration répertorié comme "app.config" dans l'explorateur de la solution. Ce fichier reflète seulement les paramètres du compilateur courant. Le compilateur ne lit pas ce fichier.

4.6 Installation de l'UIP

Le Serveur FDI importe l'UIP à partir d'un Paquetage FDI.

L'installation de l'UIP est effectuée par copie de fichier seulement. L'exécutable de l'UIP ne doit pas être enregistré dans le Cache d'assemblages global. L'UIP est installé dans une structure de dossiers qui est appelée la structure de dossiers UIP. Le Client FDI doit gérer la structure de dossiers UIP. La structure de dossiers UIP doit séparer les variantes de l'UIP l'une de l'autre afin d'éviter les conflits de nom de fichier. Les exécutables de l'UIP doivent être installés sur un chemin qui permet l'accès en lecture et en écriture pour l'exploration.

Puisque le Client FDI gère la structure de dossiers, l'UIP ne doit effectuer aucun accès à un chemin absolu. Tout accès à un fichier doit être fait par rapport à la racine d'installation de l'UIP.

Conformément à la gestion de version décrite dans l'IEC 62769-4, la coexistence d'UIP de même type, de versions majeures modifiées doit être pris en charge. Cela doit être fait en installant un UIP plus récent dans un dossier séparé. La règle "nom fort" assure que les assemblages connexes peuvent coexister pendant la durée d'exécution.

---

[1] Visual Studio est l'appellation commerciale d'un produit distribué par Microsoft Corporation. Cette information est donnée à l'intention des utilisateurs de la présente partie de l'IEC 62769 et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné. La conformité n'exige pas l'utilisation de l'appellation commerciale. L'utilisation de l'appellation commerciale exige l'autorisation du détenteur de l'appellation commerciale.

La mise en œuvre du Client FDI assure que le déploiement de l'UIP fonctionne indépendamment des authentifiants de l'utilisateur actuel. Voir la NOTE ci-dessous.

NOTE Certains dossiers gérés par le système d'exploitation exigent des droits d'accès spécifiques, par exemple, des modifications dans le dossier "Programmes" nécessitent des droits "Administrateur". Le système d'exploitation Windows fournit plusieurs moyens permettant à une application en cours d'exécution avec des droits utilisateur limités d'exécuter des actions avec des privilèges administrateur transparents à l'utilisateur, par exemple, traitement de restrictions particulières pour des répertoires identifiés, services avec des droits d'administration, exécutables qui sont configurés pour s'exécuter automatiquement avec des droits d'administration. L'alternative est de copier les exécutables de l'UIP dans des dossiers accessibles en écriture pour les utilisateurs "ordinaires".

4.7 Cycle de vie de l'UIP

4.7.1 Généralités

Le diagramme d'états de l'UIP, décrit dans l'IEC 62769-4, est composé des états Loaded (chargé), Created (créé), Operational (opérationnel), Deactivated (désactivé) et Disposed (mis au rebut). Les mécanismes qui affectent les changements d'états sont décrits dans 4.7.

Après que le Client FDI a stocké l'exécutable de l'UIP sur le Client FDI, le Client FDI charge les assemblages UIP dynamiquement dans la mémoire et exécute la logique connexe en appelant les fonctions de l'interface FDI spécifiée.

Le Paragraphe 4.7 décrit les règles relatives à la manière dont le Client FDI doit activer et désactiver l'UIP.

4.7.2 Etapes d'activation de l'assemblage UIP

4.7.2.1 Load (charger)

Le Client FDI doit charger les exécutables de l'UIP en utilisant le mécanisme LoadFrom. Le .NET Framework offre le System.Reflection.Assembly.LoadFrom à cet effet:

Le mécanisme LoadFrom se comporte comme suit.

- Le LoadFrom charge l'assemblage adressé avec le chemin du fichier et aussi les assemblages référencés situés dans le même répertoire. La chaîne d'argument assemblyFile doit contenir le nom du fichier de l'exécutable de l'UIP. Le nom du fichier de l'exécutable de l'UIP représente la fonction StartElementName décrite dans l'IEC 62769-4.
- Si un assemblage est chargé avec LoadFrom, et plus tard un assemblage dans le "contexte de chargement" tente de charger le même assemblage par le nom d'affichage, alors cette tentative de chargement échoue.
- Si un assemblage avec la même identité est déjà chargé (par exemple, par un autre UIP), alors LoadFrom renvoie l'assemblage qui a été chargé précédemment, même si un chemin de fichier différent a été spécifié. Même un nom du fichier différent importe peu. Seule l'identité de l'assemblage est importante.
- Si un assemblage est chargé avec LoadFrom, et le chemin de vérification inclut un assemblage avec la même identité (par exemple, dans le Cache d'assemblages global ou dans un répertoire de l'application), cet assemblage est alors chargé, même si un chemin de fichier différent a été spécifié.
- `LoadFrom` nécessite les permissions `FileIOPermissionAccess.Read` et `FileIOPermissionAccess.PathDiscovery` ou `WebPermission`, sur le chemin spécifié.
- `LoadFrom` charge l'assemblage dans le Domaine d'application par défaut.
- Si une image native d'un assemblage (générée par ngen.exe) existe pour le chemin d'accès spécifié, elle n'est pas alors utilisée. L'assemblage ne peut pas être chargé comme indépendant du domaine, c'est-à-dire, l'assemblage ne peut pas être partagé entre les Domaines d'application.

Ce comportement impose des règles de déploiement comme suit.

- Règles relatives aux dépendances des assemblages (voir 4.7.2.4.2).

Le Client FDI ne doit utiliser que `LoadFrom`. L'utilisation d'autres moyens de chargement d'assemblages .NET/de création d'objets n'est pas permise.

- Règles relatives aux assemblages partagés (voir 4.7.2.4.3).
- Un code machine spécifique à un processeur précompilé ne peut pas être utilisé.
- Les aspects de sécurité concernant le chargement et l'exécution des assemblages sont décrits en 4.9.

4.7.2.2 Create (créer)

La création d'une instance de l'assemblage UIP fonctionne en utilisant les fonctions de bibliothèque .NET `System.Reflection.Assembly.GetTypes` et `System.Activator.CreateInstance`. La bibliothèque de types FDI déclare un "attribut personnalisé" dénommé `UIPActivationClass`. Cet attribut doit seulement être ajouté à l'objet mettant en œuvre l'interface `IDtmUiFunction` qui en fait met en œuvre la fonction start-up de l'UIP. L'attribut `UIPActivationClass` doit être utilisé seulement une fois.

Le Client FDI peut maintenant utiliser les services `System.Reflection` afin de déterminer clairement la procédure d'activation mise en œuvre de l'UIP.

NOTE 1 La fonction System.Reflection.Assembly.GetTypes peut être utilisée pour interroger l'interface IDtmUiFunction.

NOTE 2 La fonction System.Attribute.GetCustomAttributes peut être utilisée pour lire les attributs personnalisés supplémentaires.

NOTE 3 Le résultat de l'invocation de fonction System.Activator.CreateInstance est un objet de type IDtmUiFunction.

Une diffusion de type de données est nécessaire.

4.7.2.3 Activate (activer)

L'invocation de la fonction `IDtmUiFunction.Init` active finalement l'UIP pour l'utilisateur.

4.7.2.4 Bibliothèques externes

4.7.2.4.1 Généralités

Les assemblages UIP peuvent dépendre des bibliothèques externes (bibliothèques tierces) et d'autres assemblages, par exemple, des bibliothèques spécifiques de contrôle de l'utilisateur. Les Clients FDI n'effectuent pas d'installation d'UIP, mais ils chargent dynamiquement et exécutent l'UIP. Pour prendre en charge cet usage, ainsi que l'exigence de prévenir les problèmes éventuels des assemblages en conflit, des règles sont spécifiées pour les bibliothèques externes.

Les bibliothèques externes:

- doivent être contenues à l'intérieur du Paquetage FDI;
- ne doivent pas exiger d'installation de l'installateur Microsoft (MSI);
- ne doivent pas nécessiter d'entrées dans le registre Windows ou dans le Cache d'assemblages global;
- doivent adhérer aux restrictions d'accès décrites en 4.9.2;
- doivent être compatibles aux plates-formes décrites en 4.1.2.

4.7.2.4.2 Chargement de librairies externes

Le Client FDI charge l'assemblage UIP contenant la classe "main" de l'UIP mettant en œuvre l'interface `IDtmUiFunction`, par invocation de la fonction `LoadFrom` du .NET Framework. Les assemblages référencés qui sont stockés dans le même répertoire sont automatiquement chargés ensemble avec cet assemblage .NET. Les assemblages référencés qui sont stockés dans d'autres emplacements (par exemple, dans un sous-répertoire) doivent être chargés explicitement par l'UIP lui-même.

L'UIP doit charger ces assemblages également par invocation de la fonction `LoadFrom` du .NET Framework. Le chargement des assemblages avec d'autres méthodes .NET Framework n'est pas permis.

L'usage de bibliothèques externes ne doit pas briser l'exigence d'autonomie pour les Paquetages FDI; toutes les bibliothèques externes doivent être incluses dans le Paquetage FDI de l'UIP.

4.7.2.4.3 Chargement de librairies externes partagées

Une bibliothèque externe est une bibliothèque externe partagée si l'identité d'un assemblage .NET connexe peut être utilisée par différents exécutables de l'UIP. L'identité d'un assemblage .NET importe. Le chemin d'installation et le nom du fichier de l'assemblage ne sont pas pertinents.

L'usage de bibliothèques partagées ne doit pas briser l'exigence d'autonomie pour les Paquetages FDI. Chacun des Paquetages FDI délivrés doit être livré avec toutes les bibliothèques UIP connexes nécessaires. Le mécanisme de partage vient du mécanisme d'optimisation mis en œuvre du .NET Framework.

Si un assemblage partagé est utilisé, les règles suivantes s'appliquent.

- Toute modification incompatible à l'assemblage partagé doit conduire à une nouvelle identité, par exemple, différent numéro de version.
- Les assemblages partagés ne doivent pas présumer être chargés à partir d'un chemin d'installation spécifique, par exemple, reposer que certains fichiers sont stockés dans le même répertoire ou dans un sous-répertoire.
- Les variables statiques dans les assemblages partagés sont également partagées si l'assemblage est chargé dans le même Domaine d'application. Ainsi, les variables statiques ne doivent pas avoir d'effets secondaires dans de tels scénarios. Les bibliothèques partagées externes ne doivent pas déclarer de variables statiques.
- En raison de la règle d'autonomie définie pour le Paquetage FDI, les assemblages partagés doivent être déployés avec tous les Paquetage FDI utilisant un assemblage partagé.

4.7.2.5 Invocation du constructeur de l'UIP

La mise en œuvre du constructeur et du destructeur ne doit pas générer d'exceptions. La logique du constructeur doit être limitée pour instancier l'objet en termes de structure de données internes. La logique du destructeur doit être limitée pour détruire l'objet en termes de ressources de mémoire de publication. Le constructeur et le destructeur ne doivent pas:

- Invoquer le rappel au Client FDI.
- Invoquer d'interaction utilisateur.

4.7.3 Etapes de désactivation de l'assemblage UIP

4.7.3.1 Deactivate (désactiver)

Pour la désactivation de l'UIP, le Client FDI doit appeler l'interface `IDtmUiFunction.BeginClose` et `IDtmUiFunction.EndClose`. Suite à une exécution réussie, l'UIP doit libérer toutes les ressources et le Client FDI doit supprimer toutes les références vers l'instance de l'UIP. Le nettoyeur .NET libère enfin l'objet exécutable de l'UIP.

4.7.3.2 Dispose (mettre au rebut)

Un assemblage .NET qui est chargé dans un processus, respectivement dans le `ApplicationDomain` lié n'est jamais déchargé, sauf si le `ApplicationDomain` lui-même est détruit. Cela signifie que si le Client FDI charge un assemblage UIP dans le `ApplicationDomain` par défaut, ces assemblages et tous les assemblages dépendants ne sont jamais déchargés à moins que l'application soit fermée.

Les assemblages UIP doivent être développés avec ce comportement .NET Framework en esprit. Pour réduire la consommation de mémoire, les règles suivantes s'appliquent.

- Minimiser l'utilisation des variables statiques, car celles-ci augmentent la consommation de la mémoire de l'assemblage.
- Déplacer la fonctionnalité de l'UIP qui n'est pas toujours (ou rarement) nécessaire pour séparer les assemblages. Alors, ces assemblages ne sont qu'automatiquement ou manuellement chargés lorsque le code correspondant est exécuté.
- Utiliser les assemblages partagés chaque fois que possible.
- Le Client FDI peut exécuter les assemblages .NET dans un Domaine d'application séparé afin d'avoir la capacité de les décharger.

4.8 Interaction entre un Client FDI et un UIP

4.8.1 Traitement des éléments normalisés de l'interface utilisateur

Les UIP doivent déléguer au Client FDI la présentation et le traitement des éléments normalisés de l'interface utilisateur. Les éléments normalisés de l'interface utilisateur sont

- les actions de l'interface utilisateur avec la sémantique normalisée (Appliquer/Fermer/Aide en ligne) et
- les informations de statut spécifiques à l'UIP.

Pour assurer des interactions cohérentes de l'interface utilisateur à travers les UIP provenant de différents vendeurs, un UIP peut déléguer au Client FDI la présentation et le traitement d'actions supplémentaires spécifiques à l'UIP. Néanmoins, les UIP peuvent mettre en œuvre des actions non normalisées de l'interface utilisateur dans leur propre zone de l'interface utilisateur.

L'ensemble des actions normalisées de l'interface utilisateur et de leur sémantique respective est fixe. Cependant, la disponibilité de ces actions peut changer à tout moment en fonction de l'état interne de l'UIP. L'ensemble des actions supplémentaires spécifiques à l'UIP et de leur disponibilité individuelle n'est pas fixe. Un UIP peut ajouter, supprimer, renommer, activer ou désactiver les actions spécifiques à l'UIP à tout moment en fonction de ses exigences. L'UIP doit informer le Client FDI chaque fois que la disponibilité de ses actions normalisées ou des actions spécifiques à l'UIP change (voir les événements `IStandardActions.StandardActionItemSetChanged` et `IApplicationSpecificActions.ApplicationSpecificActionItemSetChanged`).

Un Client FDI peut utiliser des éléments dédiés de l'interface utilisateur, par exemple les contrôles de boutons, pour fournir un accès direct aux actions normalisées, ainsi que pour les invoquer indirectement dans le cadre de l'interaction entre l'utilisateur et d'autres éléments

d'interface utilisateur du Client FDI. Le Client FDI doit toujours montrer toutes les actions personnalisées exposées par un UIP avec des éléments dédiés de l'interface utilisateur.

4.8.2 Exécution de service sans blocage

4.8.2.1 Fonctions internes du Client FDI

La mise en œuvre de la fonction Begin*OperationName* doit copier le contenu de l'Argument `asyncState` dans le membre `AsyncState` de l'objet retourné `IAsyncResult`.

La partie productive (consommatrice de temps) de la fonction nommée *OperationName* doit être exécutée dans un fil différent. La synchronisation avec le fil appelant est traitée via l'objet `AsyncWaitHandle` (classe `WaitHandle`), qui est aussi un membre de l'objet `IAsyncResult`.

Lorsque le traitement de la partie productive de la fonction nommée *OperationName* a terminé les objets `IAsyncResult`, l'attribut `IsCompleted` doit être mis à `True`. Si la valeur de l'argument `AsyncCallBack` est valable (n'est pas égale à NULL), le Client FDI notifie l'UIP par le biais du rappel.

La mise en œuvre de Cancel*OperationName* utilise l'argument `IAsyncResult` pour identifier le service qui a été démarré avec Begin*OperationName*. Si Begin*OperationName* a démarré un service OPCUA, le Client FDI doit appeler le service Cancel défini par OPCUA.

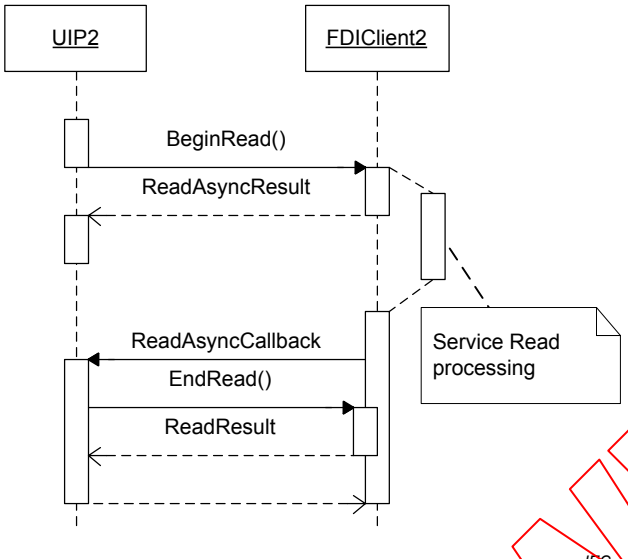
4.8.2.2 Fonctions internes de l'UIP

La gestion de plusieurs services asynchrones en parallèle doit être assurée à l'aide de l'objet `AsyncState`.

L'objet `IAsyncResult` retourné par Begin*OperationName* contient l'objet `WaitHandle`. L'UIP doit effectuer une synchronisation de fils en utilisant l'objet `WaitHandle`.

4.8.2.3 Séquence d'exécution d'un service sans blocage

Le paragraphe suivant montre la séquence d'interaction entre le Client FDI et l'UIP. Les mécanismes de gestion de fils mis en œuvre à l'intérieur du Client FDI ne sont pas représentés. L'interaction entre un Client FDI et un Serveur FDI se base sur le modèle Demande/Réponse. La demande de service du Client FDI correspond à la fonction Begin*OperationName*. L'invocation AsyncCallback correspond à la réception de la réponse du service client. End*OperationName* transmet les résultats contenus dans la réponse. La mise en œuvre de l'exécution d'un service sans blocage n'exige aucune gestion de fils à l'intérieur du Client FDI. La Figure 4 représente un exemple d'exécution de service asynchrone basé sur IAsyncPattern.

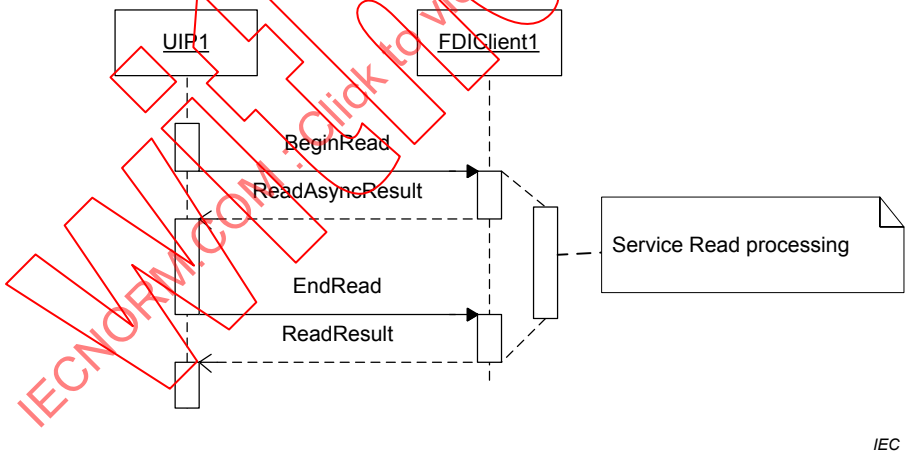


Anglais	Français
Service Read processing	Traitement du service Read

Figure 4 – Exemple d'exécution de service asynchrone basé sur IAsyncPattern

4.8.3 Exécution de service de blocage

Les interfaces fournies par le Client FDI permettent de donner un accès au Modèle d'Information synchrone en utilisant la fonctionnalité décrite en 4.8.1 de la façon montrée à la Figure 5.



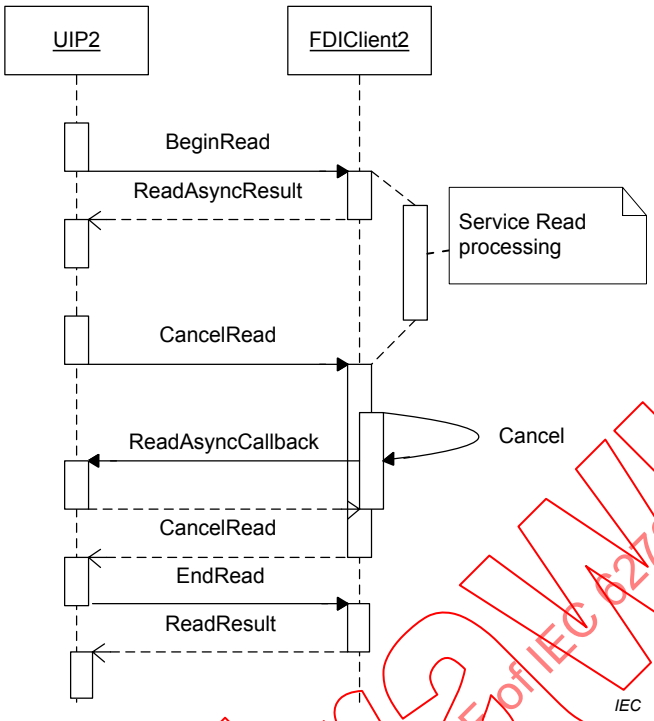
Anglais	Français
Service Read processing	Traitement du service Read

Figure 5 – Exemple d'exécution de service de blocage utilisant le modèle basé sur IAsyncResult

`ReadAsyncResult` est l'objet mettant en œuvre l'interface `IAsyncResult`.

4.8.4 Exécution du service Cancel

Certains services spécifiés pour l'interface `IDeviceModel` (voir Tableau 3) prennent en charge l'annulation d'un service débuté au moyen de la fonction Cancel*OperationName.* La Figure 6 représente la séquence de traitement sur la base de l'exemple du service "Read".



Anglais	Français
Service Read processing	Traitement du service Read
Cancel	Supprimer

Figure 6 – Exemple de séquence de traitement du service "Cancel"

L'invocation de `CancelRead` déclenche les fonctions internes au Client FDI nécessaires pour annuler l'opération de lecture active. Le Client FDI peut ne pas être en mesure d'annuler immédiatement l'opération, mais il convient qu'il l'annule aussitôt que possible. Une fois que l'opération a été annulée, le Client FDI notifie l'UIP à travers la `ReadAsyncCallback`. L'UIP doit alors appeler la fonction `EndRead`.

NOTE Un défi général mettant en œuvre ce modèle est de traiter des conditions de course correctement sur les deux côtés (UIP2 et FDIClient2). Si le Client FDI a transmis l'exécution du service via un service OPC UA, l'exécution du service effectif se déroulera à l'intérieur du Serveur FDI.

Selon la façon dont l'UIP est hébergé, il peut y avoir trois processus de travail séparés. Par conséquent, la demande d'annulation (envoyée par l'UIP) peut apparaître juste dès que le Serveur FDI a déjà terminé la demande de service. La réponse associée envoyée par le Serveur FDI peut être arrivée au Client FDI (ou pas). Le Client FDI peut appeler la fonction ReadAsyncCallBack tandis que l'UIP appelle la fonction CancelRead.

`ReadAsyncResult` est l'objet mettant en œuvre l'interface `IAsyncResult`.

4.8.5 Threading (enfilage)

4.8.5.1 Règles de mise en œuvre

L'UIP doit être en mesure de recevoir des appels dans n'importe quel fil.

L'UIP ne doit pas bloquer les appels provenant du Client FDI.

L'UIP ne doit pas utiliser le fil du Client FDI pour signaler en retour le rappel au Client FDI lui-même. Il s'agit d'éviter les interblocages et les boucles infinies.