

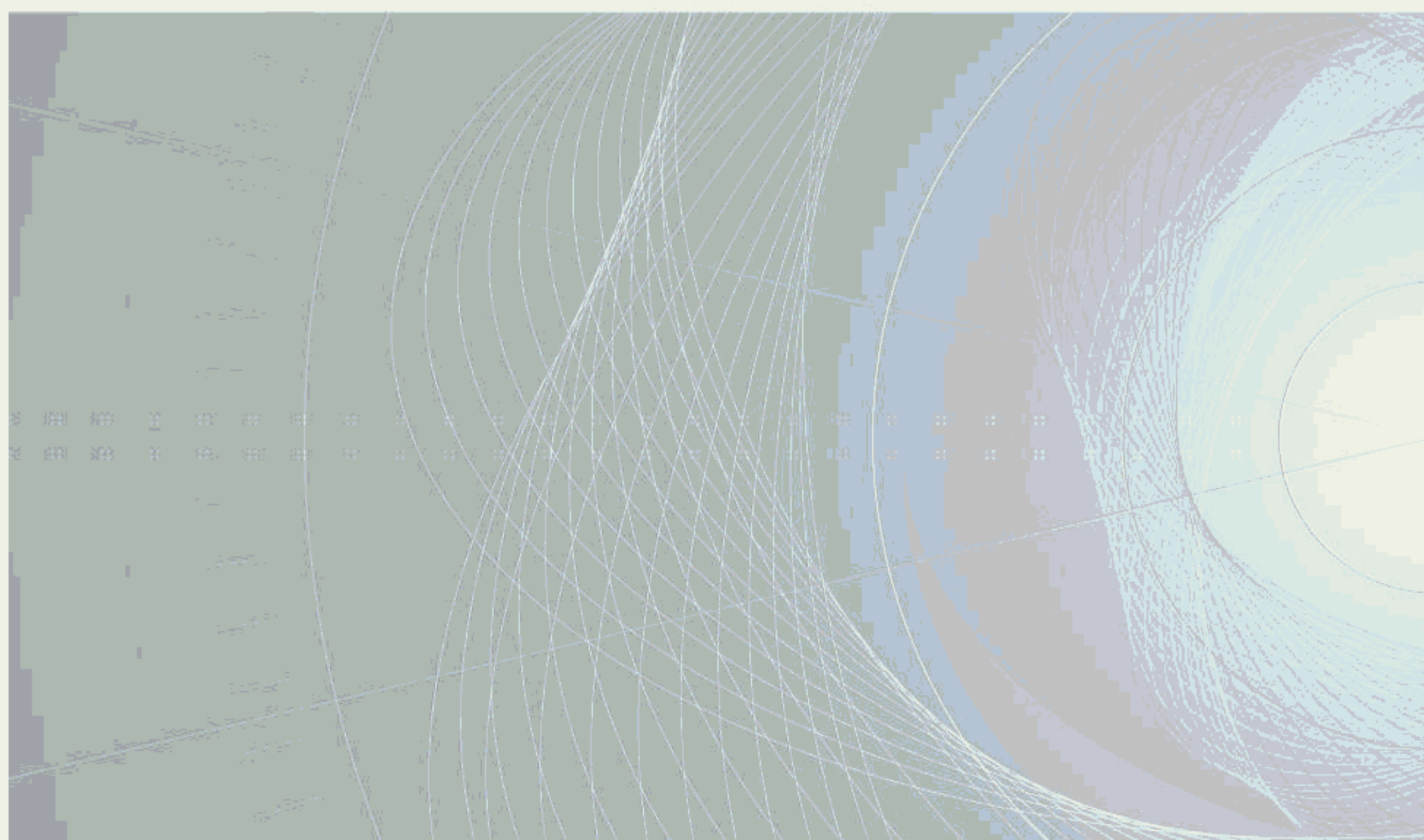
# INTERNATIONAL STANDARD

## NORME INTERNATIONALE



**Field device integration (FDI) –  
Part 6: Technology Mapping**

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







**THIS PUBLICATION IS COPYRIGHT PROTECTED**  
**Copyright © 2021 IEC, Geneva, Switzerland**

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

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

Tel.: +41 22 919 02 11  
[info@iec.ch](mailto:info@iec.ch)  
[www.iec.ch](http://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 corrigendum or an amendment might have been published.

**IEC publications search - [webstore.iec.ch/advsearchform](http://webstore.iec.ch/advsearchform)**

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 It also gives information on projects, replaced and withdrawn publications.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](http://webstore.iec.ch/justpublished)**

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

**IEC Customer Service Centre - [webstore.iec.ch/csc](http://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: [sales@iec.ch](mailto:sales@iec.ch).

**IEC online collection - [oc.iec.ch](http://oc.iec.ch)**

Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

**Electropedia - [www.electropedia.org](http://www.electropedia.org)**

The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 18 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

**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.

**Recherche de publications IEC -**

**[webstore.iec.ch/advsearchform](http://webstore.iec.ch/advsearchform)**

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](http://webstore.iec.ch/justpublished)**

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et une fois par mois par email.

**Service Clients - [webstore.iec.ch/csc](http://webstore.iec.ch/csc)**

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: [sales@iec.ch](mailto:sales@iec.ch).

**IEC online collection - [oc.iec.ch](http://oc.iec.ch)**

Découvrez notre puissant moteur de recherche et consultez gratuitement tous les aperçus des publications. Avec un abonnement, vous aurez toujours accès à un contenu à jour adapté à vos besoins.

**Electropedia - [www.electropedia.org](http://www.electropedia.org)**

Le premier dictionnaire d'électrotechnologie en ligne au monde, avec plus de 22 000 articles terminologiques en anglais et en français, ainsi que les termes équivalents dans 16 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.





IEC 62769-6

Edition 2.0 2021-02

# INTERNATIONAL STANDARD

## NORME INTERNATIONALE



---

**Field device integration (FDI) –  
Part 6: Technology Mapping**

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

INTERNATIONAL  
ELECTROTECHNICAL  
COMMISSION

COMMISSION  
ELECTROTECHNIQUE  
INTERNATIONALE

ICS 25.040.40; 35.100.05

ISBN 978-2-8322-9312-6

**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

FOREWORD .....	4
INTRODUCTION .....	6
1 Scope .....	7
2 Normative references .....	7
3 Terms, definitions, abbreviated terms, symbols and conventions .....	7
3.1 Terms and definitions .....	7
3.2 Abbreviated terms .....	8
3.3 Symbols .....	8
3.4 Conventions .....	8
4 Technical concepts .....	8
4.1 General .....	8
4.1.1 Overview .....	8
4.1.2 Platforms .....	9
4.1.3 FDI Type Library .....	9
4.2 UIP representation .....	10
4.3 UIP executable representation .....	11
4.4 UIP executable compatibility rules .....	11
4.5 Allowed .NET Common Language Run-time versions .....	11
4.5.1 General .....	11
4.5.2 CLR compatibility strategy .....	11
4.5.3 How to identify the .NET target platform of a UIP .....	12
4.6 UIP Deployment .....	12
4.7 UIP Lifecycle.....	13
4.7.1 General .....	13
4.7.2 UIP Assembly activation steps .....	13
4.7.3 UIP Assembly deactivation steps .....	15
4.8 Interaction between an FDI Client and a UIP .....	16
4.8.1 Handling of standard UI elements .....	16
4.8.2 Non-blocking service execution .....	16
4.8.3 Blocking service execution.....	17
4.8.4 Cancel service execution .....	18
4.8.5 Threading .....	19
4.8.6 Timeout .....	19
4.8.7 Exception handling .....	20
4.8.8 Type safe interfaces .....	21
4.8.9 Globalization and localization .....	21
4.8.10 WPF Control handling .....	21
4.8.11 Win Form handling .....	21
4.9 Security .....	21
4.9.1 General .....	21
4.9.2 Access permissions .....	22
4.9.3 Code identity concept .....	22
5 Interface definition .....	23
Figure 1 – FDI Type Library structure .....	10

Figure 2 – .NET surrogate process .....	12
Figure 3 – Identification of Run-time Version.....	12
Figure 4 – IAsyncPattern based asynchronous service execution example .....	17
Figure 5 – Blocking service execution example using IAsyncResult based pattern .....	18
Figure 6 – Cancel service processing sequence example .....	18
Figure 7 – Exception source .....	20
 Table 1 – Technology edition reference .....	 9
Table 2 – Base Property Services .....	23
Table 3 – Device Model Services .....	23
Table 4 – Access Control Services .....	24
Table 5 – Direct Access Services .....	24
Table 6 – Hosting Services .....	24
Table 7 – UIP Services .....	26
Table 8 – Base Data Types .....	26
Table 9 – Special Types .....	27

## INTERNATIONAL ELECTROTECHNICAL COMMISSION

---

FIELD DEVICE INTEGRATION (FDI) –

## Part 6: 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.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

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.

This second edition cancels and replaces the first edition published in 2015. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

- a) redesign of the security concept for UIP execution.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
65E/763/FDIS	65E/773/RVD

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

This document 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 document will remain unchanged until the stability date indicated on the IEC website under "<http://webstore.iec.ch>" in the data related to the specific document. At this date, the document 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 IEC 62769 series has the general title *Field Device Integration (FDI)* and the following parts:

- Part 1: Overview
- Part 2: FDI Client
- Part 3: FDI Server
- Part 4: FDI Packages
- Part 5: FDI Information Model
- Part 6: FDI Technology Mapping
- Part 7: FDI Communication Devices
- Part 100: Profiles – Generic Protocol Extensions
- Part 101-1: Profiles – Foundation Fieldbus H1
- Part 101-2: Profiles – Foundation Fieldbus HSE
- Part 103-1: Profiles – PROFIBUS
- Part 103-4: Profiles – PROFINET
- Part 109-1: Profiles – HART and WirelessHART
- Part 115-2: Profiles – Protocol-specific Definitions for Modbus RTU
- Part 150-1: Profiles – ISA 100.11a

## FIELD DEVICE INTEGRATION (FDI) –

### Part 6: 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/.NET as defined in IEC 62769-4.

## 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 61804 (all parts), *Function blocks (FB) for process control and Electronic Device Description Language (EDDL)*

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

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

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

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

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, symbols 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.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

#### 3.1.1

##### **Application Domain**

isolated environment where applications execute

### 3.1.2

#### **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

### 3.1.3

#### **Global Assembly Cache**

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

### 3.1.4

#### **Windows Registry**

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

## 3.2 Abbreviated terms

For the purposes of this document, the abbreviated terms 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 graphical symbols in accordance with ISO/IEC 19505-1 (UML 2.0).

## 3.4 Conventions

For the purposes of this document, the conventions given in IEC 62769-1 apply.

The description of Non-blocking service execution in 4.8.2 uses italics to identify a generic operation name the internal function is being applied to.

# 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 × 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	2016
OPC UA (Parts 1-8)	IEC 62541	2015
Open Packaging Convention	ISO/IEC 29500	2016
Extensible Markup Language (XML)	N/A	W3C, 1.0 (fifth edition)

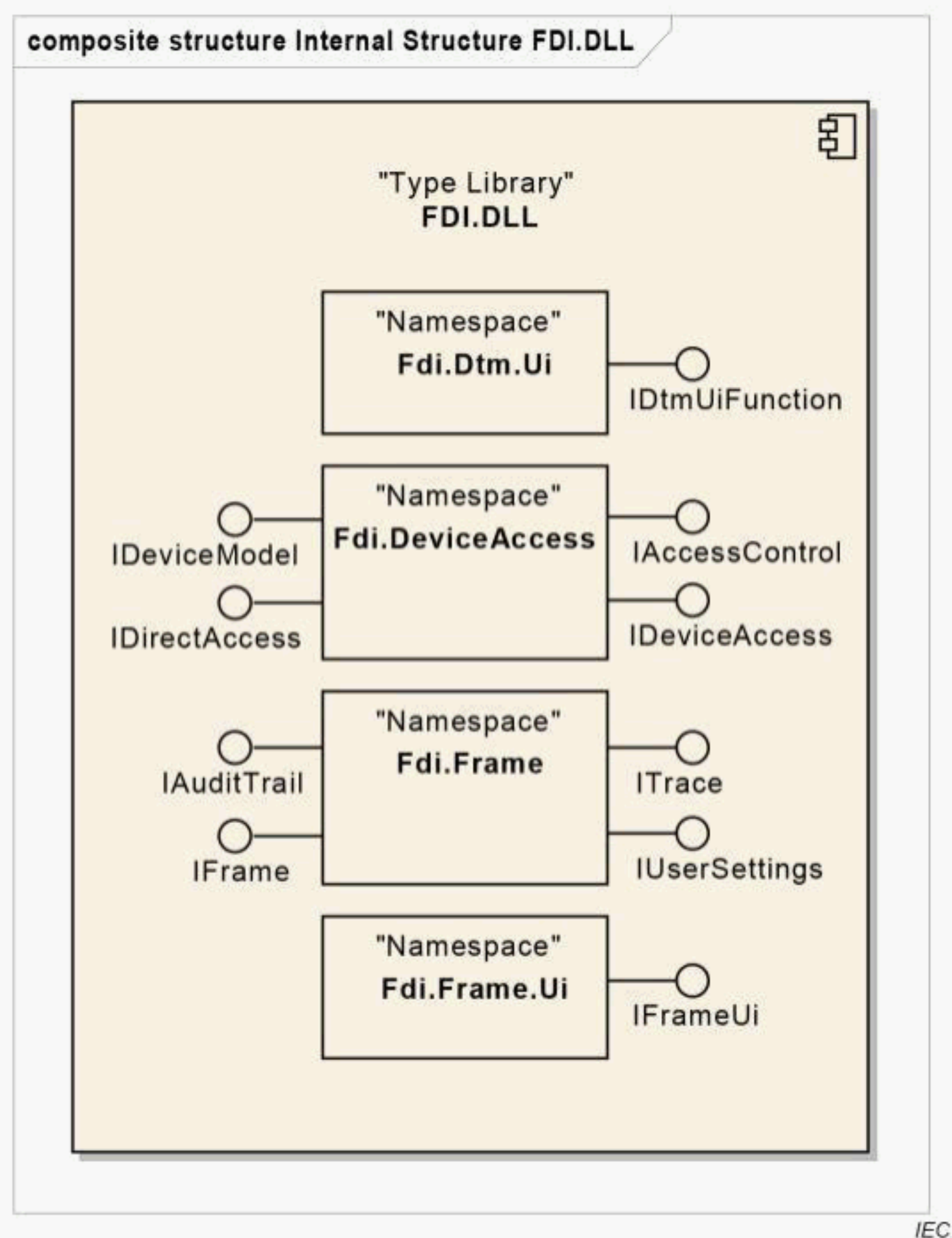
### 4.1.3 FDI Type Library

The Device Access Services and the UIP Services can be modelled 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 file name of this Assembly shall be 'fdi.dll'. The fdi.dll shall be versioned as per IEC 62769-1:2020, 8.1. The FDI Type Library is part of the FDI Core Technology as per IEC 62769-1:2020, 8.3.2.1 and therefore directly influences the FDI Technology Version. All Compatible changes of the fdi.dll lead to an increase of the minor portion of the FDI Technology Version. Incompatible changes lead to an increase of the major portion of the FDI Technology Version (see IEC 62769-1:2020, 8.3.2.2).

The FDI Type Library is signed with a single unique key by the issuer of the file. The FDI Type Library shall be installed separately as part of every FDI Client installation. User Interface Plug-Ins (UIP) and the FDI Client Application shall use this instance of the fdi.dll. UIPs shall not carry or deploy the FDI Type Library. The FDI Client is responsible to provide means to allow updates of this type library over time.

Figure 1 shows the FDI Type Library structure.



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

**Figure 1 – FDI Type Library structure**

## 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 UIP 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 Resource files and application configuration data.

The RuntimeId of a UIP Variant shall be ".NET Framework CLR4", see IEC 62769-4. FDI Clients supporting this RuntimeId shall support the .NET Framework 4.6.1 or higher using the CLR4 and UIPs with this RuntimeId shall use the .NET Framework 4.6.1 or lower supporting the CLR4 (meaning .NET Framework 4.0 up to .NET Framework 4.6.1).

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 compatibility rules for different versions of the UIP component are specified in IEC 62769-4.

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 by 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. If 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.

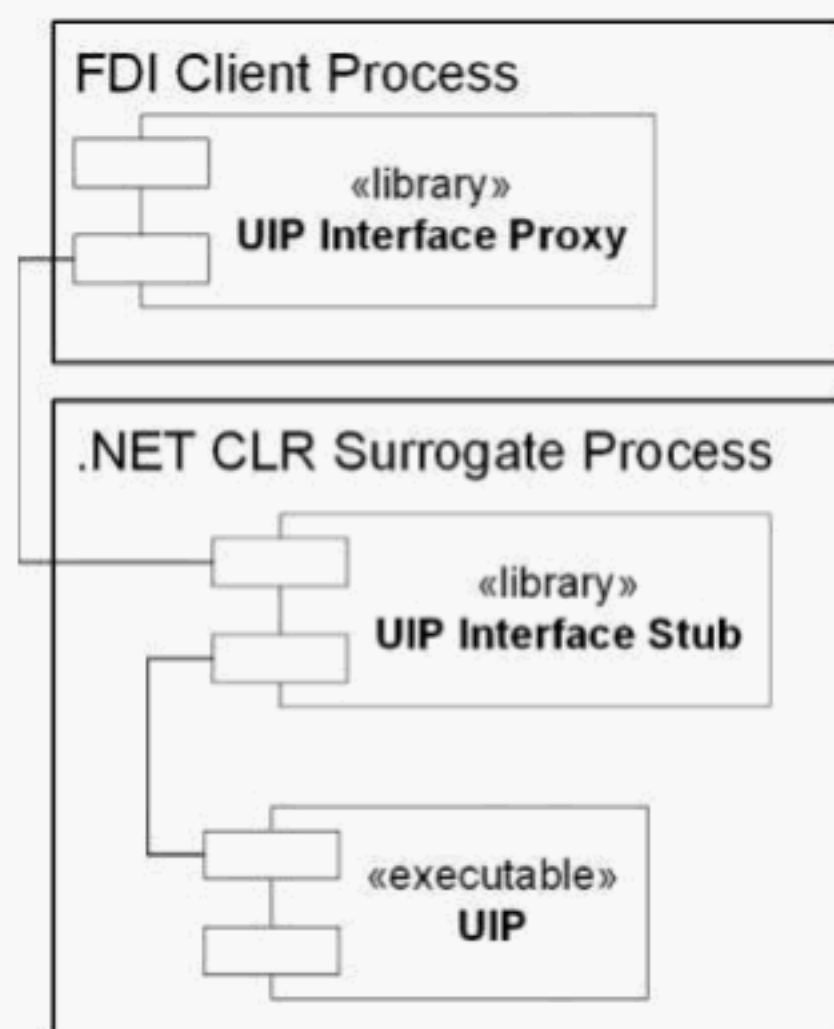
#### 4.5.2 CLR compatibility strategy

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.



IEC

**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 compiler setting. The compiler does not read that file.

#### 4.6 UIP Deployment

The general UIP installation rules are outlined in IEC 62769-2. The UIP executable shall not be registered within the Global Assembly Cache.

The "strong-name" rule ensures that related Assemblies of different versions of the UIP can coexist during runtime.

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

<sup>1</sup> Visual Studio is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of this product.

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 (third-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 `IAAsyncResult` 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 `IAAsyncResult` object.

When processing of the productive part of the function named `OperationName` has finished, the `IAAsyncResult` 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 `IAAsyncResult` 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.

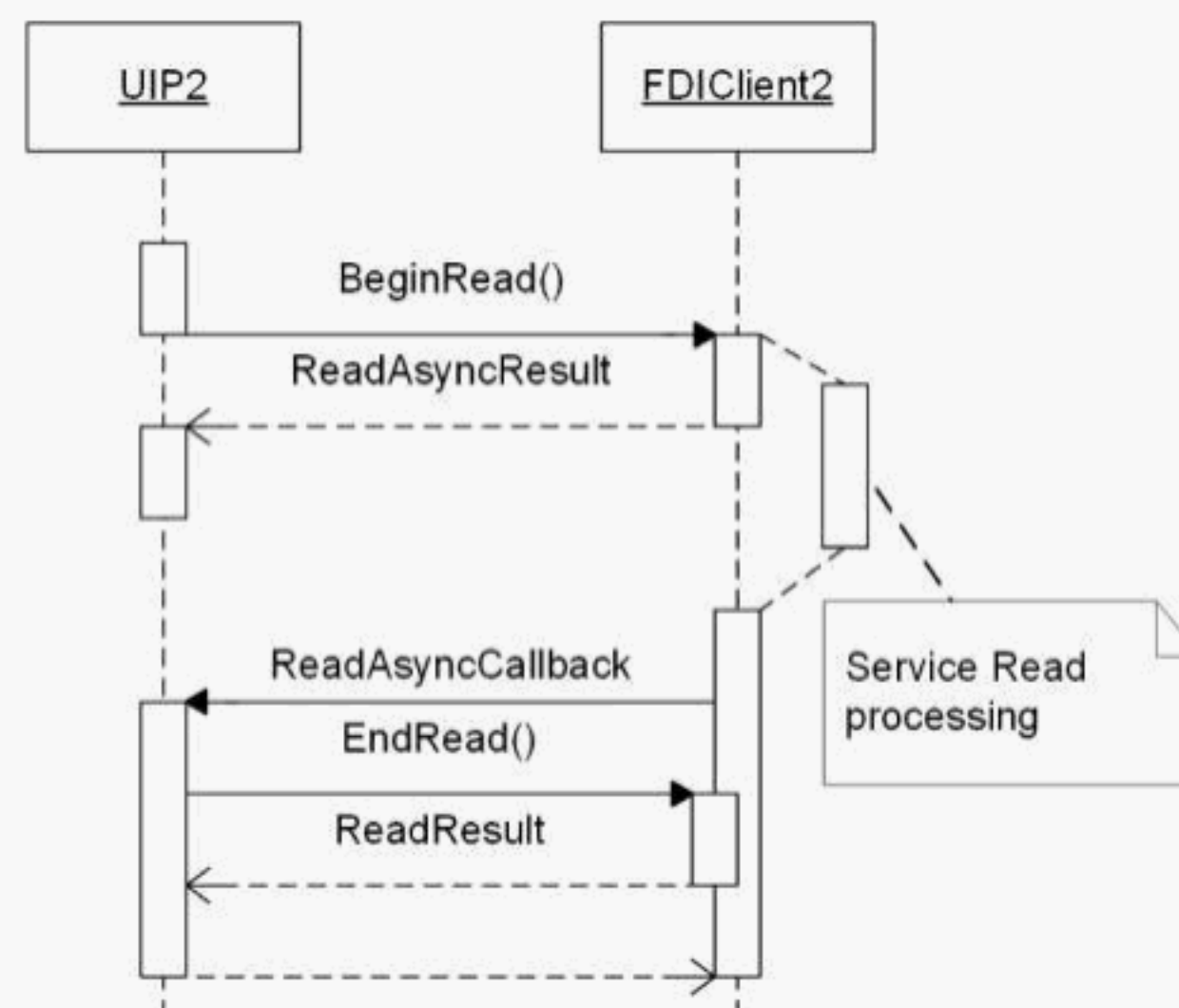
#### 4.8.2.2 UIP internal functions

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. `EndOperationName` 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 an `IAsyncPattern` based Asynchronous service execution.



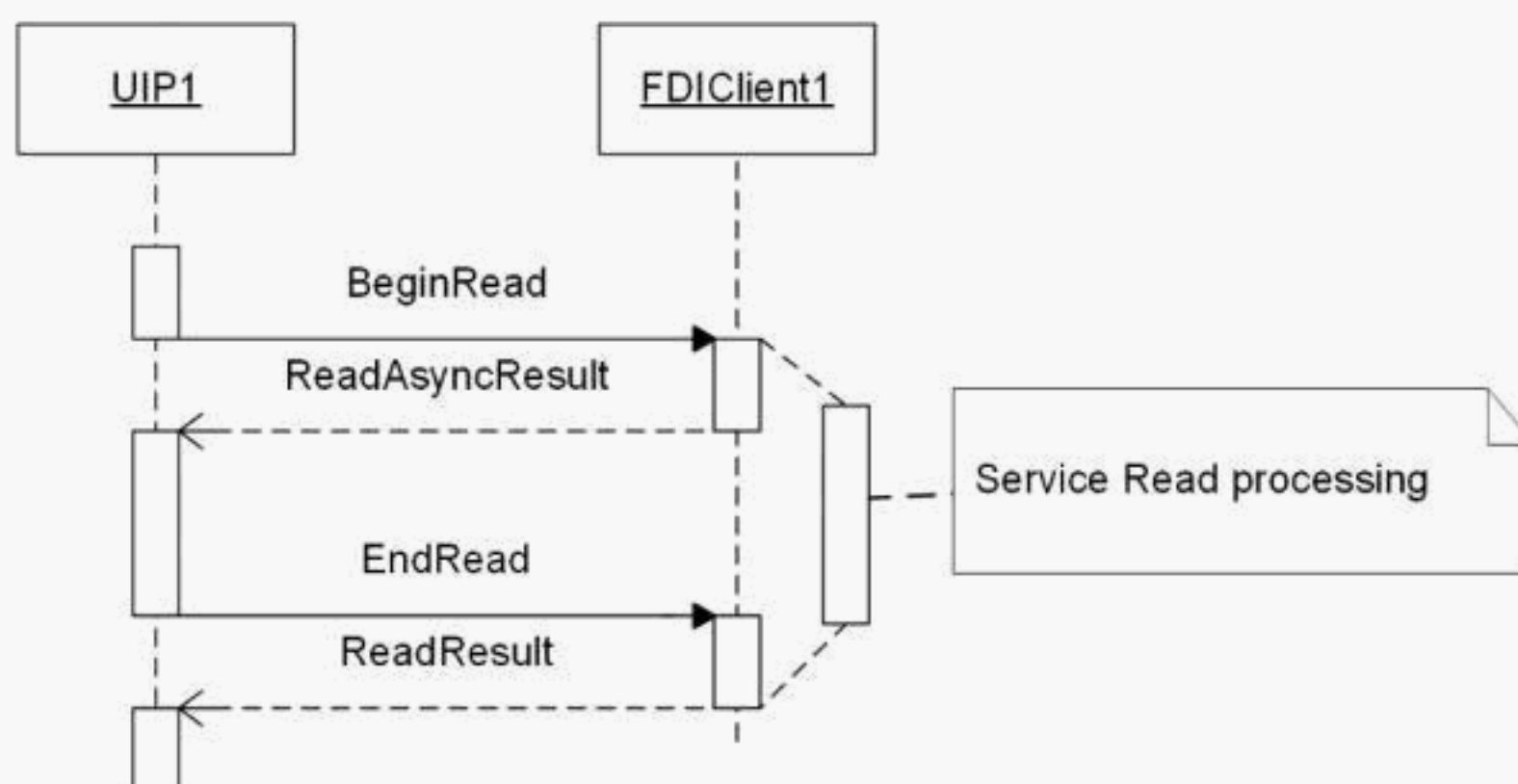
IEC

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.





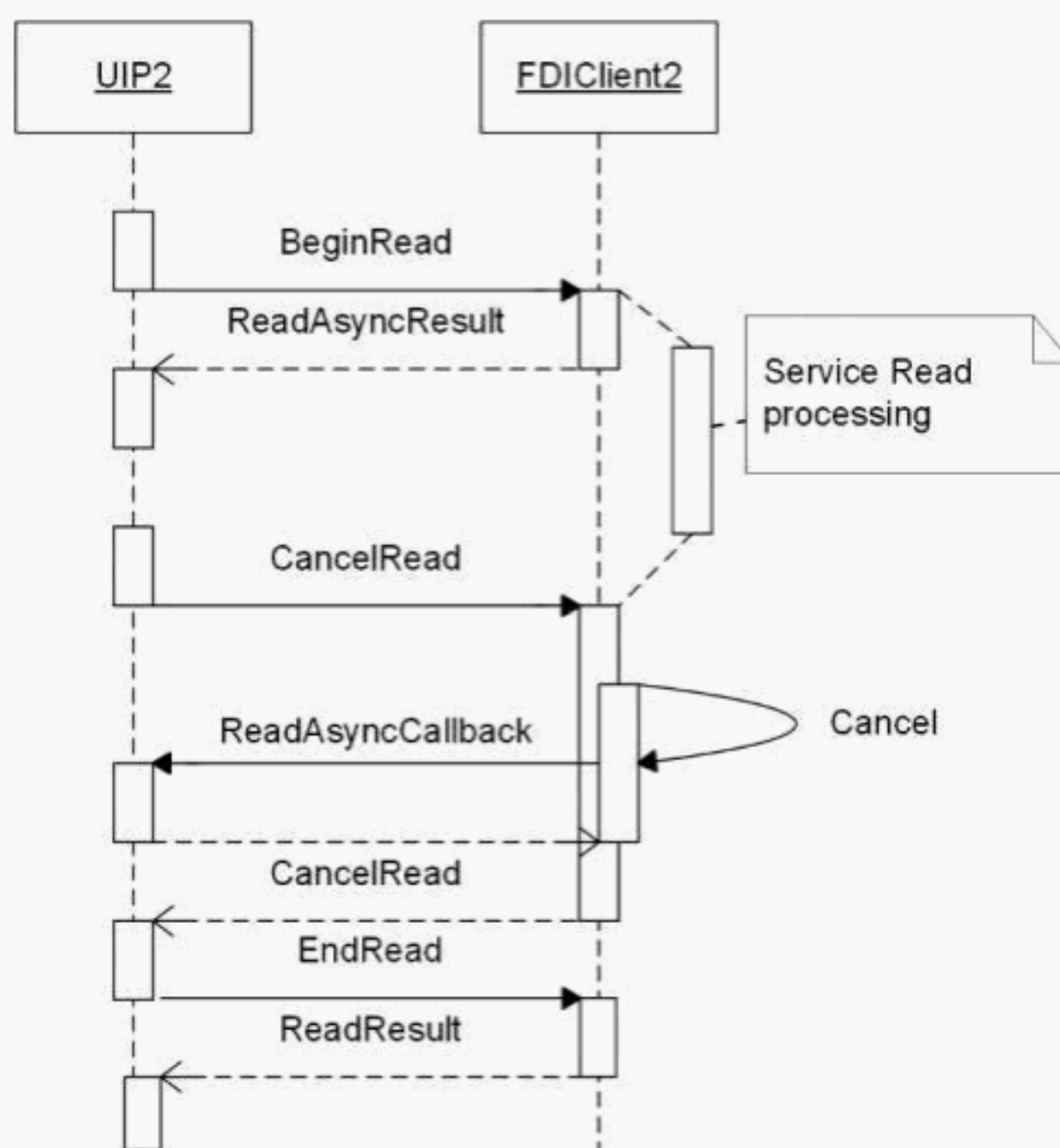
IEC

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

**Figure 5 – Blocking service execution example using `IAsyncResult` based pattern**

#### 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 OperationName`. The following Figure 6 will illustrate the processing sequence based on the Read service example.



IEC

**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 `IAAsyncResult`.

## 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 receiving 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 the 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 the 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:

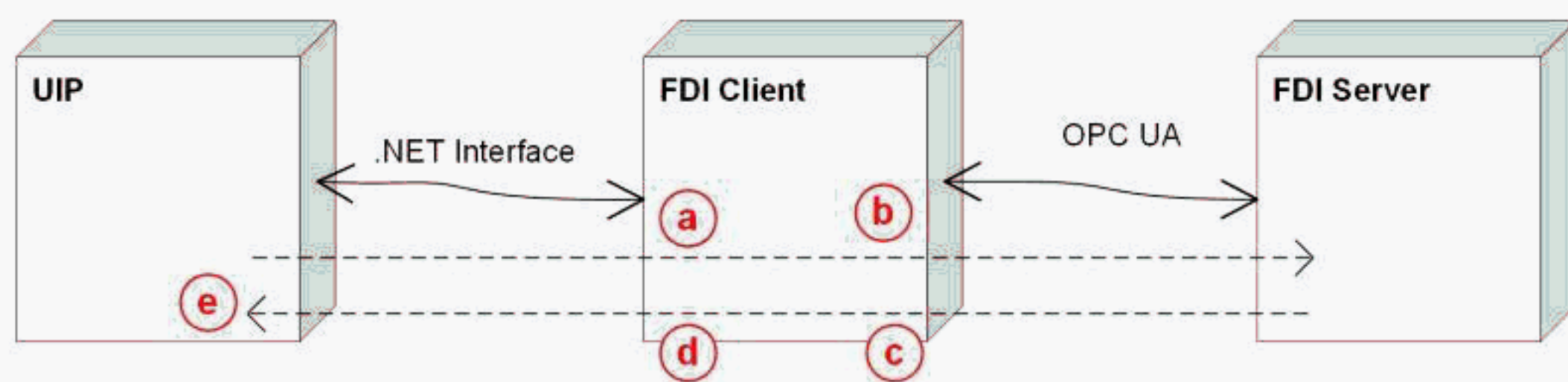
- Exceptions that indicate software states or events that have not been anticipated during the software development are considered as software quality issues (Run-time error).
- Exceptions indicating anticipated software operation failures.

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.



IEC

**Figure 7 – Exception source**

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

- passing the request from the UIP to the FDI Client;
- request forwarding inside the FDI Client;
- processing the response from the FDI Server;
- forwarding the response to the UIP;
- 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 `IAAsyncResult` 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 *EndOperationName* 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

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. Conversely, a UIP implements the accessibility function. The related rules affect, on the one hand, 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. Conversely, a UIP implements the accessibility function. The related rules affect, on the one hand, 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's stability, integrity and sensitive data.

System-wide security begins with the design process, which is out of the scope of standardization. From the system's 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 all FDI Packages need compliance testing and certification, the presumption is that such certified FDI Packages don't pose any threats to a system. This means that a UIP could be executed with fully trusted permissions.

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

## **4.9.2 Access permissions**

### **4.9.2.1 General**

Within 4.9.2, technology-specific permissions and restrictions are specified in addition to the one specified in IEC 62769-2. The permissions and restrictions shall be enforced by the FDI Client. The implementation rules define how this shall be implemented.

### **4.9.2.2 Technology specific UIP permissions and restrictions**

The general UIP permissions and restrictions are specified in IEC 62769-2. The permissions and restrictions specified in 4.9.2 are specific to .NET-based UIPs.

- a) Launching of an Active-X component is not allowed.
- b) A UIP shall not write to the operating system registry. Read access to registry is allowed.

The FDI Client shall restrict the UIP permissions in accordance with this list.

### **4.9.2.3 Implementation rules**

In order to restrict the permissions of a UIP, an FDI Client shall execute a UIP in a separate process (not the FDI Client process) that should run with a separate user account (4.5.2 describes how an FDI Client can manage hosting a UIP in a different process). By restricting the permissions of the separate user account, the FDI Client also restricts the permissions of the UIP running under the separate user account.

How the separate user account is created and configured is host specific. That includes

- whether it is a local or a domain account;
- whether it is used only for UIP hosting or also for other purposes;
- whether it is reused for all UIP processes of the same FDI Client instance or several FDI Clients;
- how restrictions are implemented (e.g. limited access to the file system by mechanism of the operating system, blocking network access by a firewall).

Whether the FDI Client executes each UIP instance in its own process, or executes several or all UIP instances in the same process, is FDI Client-specific.

As UIPs run in a separate process, UIP developers need to be aware that opening a modal dialog with operating system mechanisms will only provide modality within the Windows of that process. To avoid this behavior, UIP developers can start the modal UIP or start a new modal UIP.

## **4.9.3 Code identity concept**

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

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

- unique identification of UIP executables;
- code integrity verification.

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

## 5 Interface definition

Table 2 to Table 9 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 <sup>a)</sup>	Fdi.Frame.Ui.IFrameUi.BeginOpenDtmUiModal Fdi.Frame.Ui.IFrameUi.EndOpenDtmUiModal  Fdi.Frame.Ui.IFrameUi.BeginOpenDtmUi Fdi.Frame.Ui.IFrameUi.EndOpenDtmUi
CloseUserInterface <sup>a)</sup>	Fdi.Dtm.Ui.CloseMeRequestHandler <sup>b)</sup>  Fdi.Frame.Ui.IFrameUi.BeginCloseDtmUi Fdi.Frame.Ui.IFrameUi.EndCloseDtmUi
LogAuditTrailMessage	Fdi.Frame.IAuditTrail.Notify
SaveUserSettings	Fdi.Frame.IUserSettings.SaveUserSettings
LoadUserSettings	Fdi.Frame.IUserSettings.LoadUserSettings
Trace	Fdi.Frame.ITrace.TraceData Fdi.Frame.ITrace.TraceEvent
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



Abstract Service	.NET Implementation
ExportFile	Fdi.Frame.ImportExport.IFileExport.BeginInitExport Fdi.Frame.ImportExport.IFileExport.EndInitExport Fdi.Frame.ImportExport.IFileExport.BeginWriteExport Fdi.Frame.ImportExport.IFileExport.EndWriteExport Fdi.Frame.ImportExport.IFileExport.BeginFinishExport Fdi.Frame.ImportExport.IFileExport.EndFinishExport
CancelExportFile	Fdi.Frame.ImportExport.IFileExport.BeginFinishExport Fdi.Frame.ImportExport.IFileExport.EndFinishExport
ImportFile	Fdi.Frame.ImportExport.IFileExport.BeginInitImport Fdi.Frame.ImportExport.IFileExport.EndInitImport Fdi.Frame.ImportExport.IFileExport.BeginReadImport Fdi.Frame.ImportExport.IFileExport.EndReadImport Fdi.Frame.ImportExport.IFileExport.BeginFinishImport Fdi.Frame.ImportExport.IFileExport.EndFinishImport
CancellImportFile	Fdi.Frame.ImportExport.IFileExport.BeginFinishImport Fdi.Frame.ImportExport.IFileExport.EndFinishImport
OpenDefaultApplication	Fdi.Frame.DefaultApplication.IOpenDefaultApplication.BeginInitOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.EndInitOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.BeginWriteOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.EndWriteOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.BeginFinishOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.EndFinishOpenDefaultApplication
GetEnvironmentProperties	Fdi.Frame.HostProperty. GetHostingProperties

- a) Functions `OpenUserInterface`, `CloseUserInterface` shall only be started using the operation pattern described in 4.8.2.3.
- b) 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	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
ApplicationSpecificActionItemsChangeCallba ck	Fdi.Dtm.Ui. IApplicationSpecificActions.ApplicationSpecificActionItemSetChan ged
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)



## SOMMAIRE

AVANT-PROPOS .....	30
INTRODUCTION .....	32
1 Domaine d'application .....	33
2 Références normatives .....	33
3 Termes, définitions, termes abrégés, symboles et conventions .....	33
3.1 Termes et définitions .....	33
3.2 Termes abrégés .....	34
3.3 Symboles .....	34
3.4 Conventions .....	34
4 Concepts techniques .....	34
4.1 Généralités .....	34
4.1.1 Vue d'ensemble .....	34
4.1.2 Plates-formes .....	34
4.1.3 Bibliothèque de Types FDI .....	35
4.2 Représentation de l'UIP .....	36
4.3 Représentation de l'exécutable de l'UIP .....	36
4.4 Règles de compatibilité de l'exécutable de l'UIP .....	37
4.5 Versions permises du CLR (Common Language Run-time) .NET .....	37
4.5.1 Généralités .....	37
4.5.2 Stratégie de compatibilité CLR .....	37
4.5.3 Comment identifier la plate-forme cible .NET d'un UIP .....	38
4.6 Déploiement de l'UIP .....	38
4.7 Cycle de vie de l'UIP .....	39
4.7.1 Généralités .....	39
4.7.2 Étapes d'activation de l'Assemblage UIP .....	39
4.7.3 Étapes de désactivation de l'Assemblage UIP .....	41
4.8 Interaction entre un Client FDI et un UIP .....	42
4.8.1 Traitement des éléments normalisés de l'Interface Utilisateur .....	42
4.8.2 Exécution d'un service sans blocage .....	42
4.8.3 Exécution d'un service de blocage .....	43
4.8.4 Exécution du service Cancel .....	43
4.8.5 Threading (enfilage) .....	44
4.8.6 Expiration de délai .....	45
4.8.7 Traitement des exceptions .....	45
4.8.8 Interfaces de type sûr (type safe) .....	46
4.8.9 Globalisation et localisation .....	46
4.8.10 Traitement de contrôle WPF .....	47
4.8.11 Traitement des formes de Windows .....	47
4.9 Sécurité .....	47
4.9.1 Généralités .....	47
4.9.2 Permissions d'accès .....	47
4.9.3 Concept d'identité de code .....	48
5 Définition d'interface .....	48
Figure 1 – Structure de la Bibliothèque de Types FDI .....	36

Figure 2 – Processus de substitution .NET .....	38
Figure 3 – Identification de la version exécutable .....	38
Figure 4 – Exemple d'exécution d'un service asynchrone fondé sur IAsyncPattern.....	43
Figure 5 – Exemple d'exécution d'un service de blocage avec le modèle fondé sur IAsyncResult .....	43
Figure 6 – Exemple de séquence de traitement du service "Cancel" .....	44
Figure 7 – Source d'exception .....	46
Tableau 1 – Référence de l'édition de technologie .....	35
Tableau 2 – Services de Propriété de Base .....	49
Tableau 3 – Services de Modèle d'Appareil .....	49
Tableau 4 – Services de Contrôle d'Accès .....	49
Tableau 5 – Services d'Accès Direct .....	49
Tableau 6 – Services d'Hébergement .....	50
Tableau 7 – Services d'UIP .....	51
Tableau 8 – Types de données de base .....	52
Tableau 9 – Types particuliers .....	52

# COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

## INTÉGRATION DES APPAREILS DE TERRAIN (FDI) –

### Partie 6: Mapping de technologies

#### AVANT-PROPOS

- 1) La Commission Électrotechnique 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. À 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.
- 9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 62769-6 a été établie par le sous-comité 65E: Les dispositifs 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.

Cette deuxième édition annule et remplace la première édition parue en 2015. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

- a) redéfinition du concept de sécurité pour l'exécution de l'UIP.



Le texte de cette Norme internationale est issu des documents suivants:

FDIS	Rapport de vote
65E/763/FDIS	65E/773/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 internationale.

Ce document a été rédigé 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 ce document 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 au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

**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 série IEC 62769 est publiée sous le titre général " *Intégration des appareils de terrain (FDI)* " et comporte les parties suivantes:

- Partie 1: Vue d'ensemble
- Partie 2: Client FDI
- Partie 3: Serveur FDI
- Partie 4: Paquetages FDI
- Partie 5: Modèle d'Information FDI
- Partie 6: Mapping de technologies FDI
- Partie 7: Appareils de Communication FDI
- Partie 100: Profils – Extensions de protocoles génériques
- Partie 101-1: Profils – Foundation Fieldbus H1
- Partie 101-2: Profils – Foundation Fieldbus HSE
- Partie 103-1: Profils – PROFIBUS
- Partie 103-4: Profils – PROFINET
- Partie 109-1: Profils – HART et WirelessHART
- Partie 115-2: Profils – Définitions spécifiques au protocole pour Modbus-RTU
- Partie 150-1: Profils – ISA 100.11a

## INTÉGRATION DES APPAREILS DE TERRAIN (FDI) –

### Partie 6: Mapping de technologies

#### 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 porte essentiellement 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 WORKSTATION (Poste de travail)/.NET telle que définie dans l'IEC 62769-4.

#### 2 Références normatives

Les documents ci-après, dans leur intégralité ou non, sont des références normatives indispensables à l'application du présent document. 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 61804 (toutes les parties), *Blocs fonctionnels (FB) pour les procédés industriels et le Langage de Description Electronique de Produit (EDDL)*

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

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

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

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

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, termes abrégés, symboles 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.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse <http://www.electropedia.org/>
- ISO Online browsing platform: disponible à l'adresse <http://www.iso.org/obp>

##### 3.1.1

###### Domaine d'Application

environnement isolé dans lequel s'exécutent les applications

##### 3.1.2

###### Bibliothèque de Types FDI

assemblage qui contient 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

### 3.1.3

#### **Cache Global d'Assemblages**

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

### 3.1.4

#### **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 Termes abrégés

Pour les besoins du présent document, les termes abrégés 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 des symboles graphiques conformément à l'ISO/IEC 19505-1 (UML 2.0).

## 3.4 Conventions

Pour les besoins du présent document, les conventions de l'IEC 62769-1 s'appliquent.

La description de l'exécution d'un service sans blocage en 4.8.2 utilise des caractères italiques pour identifier le nom d'une opération générique à laquelle la fonction interne est appliquée.

# 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 tout d'abord la base de la technologie pour la mise en œuvre de l'UIP, l'environnement matériel et logiciel, y compris les règles connexes de mise en œuvre. 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 connexes de mise en œuvre 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 sont spécifiées 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 le poste de travail est le suivant: une souris, un clavier et un écran couleur de résolution 1 024 × 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	Édition
.NET	N/A	CLR4 pour la mise en œuvre de l'UIP
EDDL	IEC 61804/IEC 61804	2016
OPC UA (Parties 1-8)	IEC 62541	2015
Convention de Paquetage Ouvert (OPC)	ISO/IEC 29500	2016
Langage de balisage extensible (XML)	N/A	W3C, 1.0 (cinquième édition)

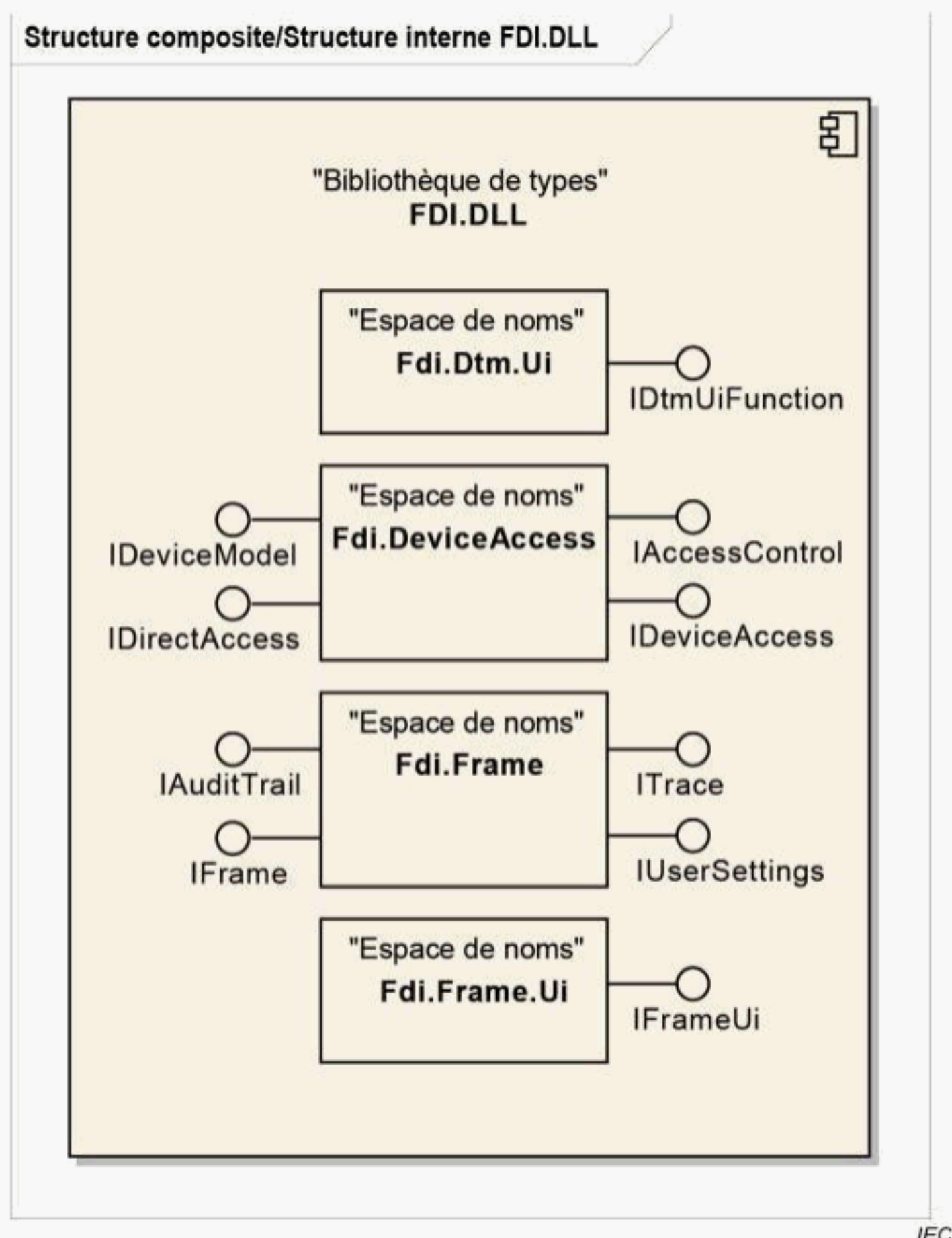
### 4.1.3 Bibliothèque de Types FDI

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. Le nom de fichier de cet Assemblage doit être 'fdi.dll'. Le fichier fdi.dll doit être versionné conformément à l'IEC 62769-1:2020, 8.1. La Bibliothèque de Types FDI fait partie de la Technique de Base FDI conformément à l'IEC 62769-1:2020, 8.3.2.1. Par conséquent, elle influe directement sur la Version de Technologie FDI. Toutes les modifications compatibles du fichier fdi.dll se traduisent par une augmentation de la portion mineure de la Version de Technologie FDI. Les modifications incompatibles se traduisent par une augmentation de la portion majeure de la Version de Technologie FDI (voir l'IEC 62769-1:2020, 8.3.2.2).

La Bibliothèque de Types FDI est signée avec une clé unique par l'émetteur du fichier. La Bibliothèque de Types FDI doit être installée séparément dans le cadre de chaque installation de Client FDI. Des Plugiciels d'Interface Utilisateur (UIP) et l'Application du Client FDI doivent utiliser cette instance du fichier fdi.dll. Les UIP ne doivent pas transporter ou déployer la Bibliothèque de Types FDI. Il incombe au Client FDI de fournir des moyens d'autoriser les mises à jour de cette bibliothèque de types au fil du temps.

La Figure 1 représente la structure de Bibliothèque de Types FDI.



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

**Figure 1 – Structure de la Bibliothèque de Types FDI**

## 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 ou les fichiers supplémentaires de la Variante de l'UIP sont appelés "suppléments de l'UIP".

Le ou les suppléments de l'UIP sont stockés dans un ou plusieurs sous-dossiers du répertoire d'installation de l'exécutable de l'UIP.

EXEMPLE Fichiers de ressources et données de configuration d'application.

Le RuntimeId d'une Variante de l'UIP doit être ".NET Framework CLR4", voir l'IEC 62769-4. Les Clients FDI qui prennent en charge ce RuntimeId doivent prendre en charge le .NET Framework 4.6.1 ou supérieur utilisant le CLR4 et les UIP avec ce RuntimeId doivent utiliser le .NET Framework 4.6.1 ou inférieur qui prend en charge le CLR4 (c'est-à-dire: de .NET Framework 4.0 jusqu'à .NET Framework 4.6.1).

La Variante de l'UIP doit être autonome. Toutes les bibliothèques exigées 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 exigées doivent avoir des noms forts. La signature d'un Assemblage ayant un nom fort peut être effectuée au moyen d'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 exigées doivent être livrés avec le fichier qui contient la clé publique afin de permettre la vérification de l'Assemblage.

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

Les règles de compatibilité pour les différentes versions du composant de l'UIP sont spécifiées dans l'IEC 62769-4.

La plate-forme cible de compilation pour l'UIP doit être "anyCPU". Si cette plate-forme 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 en mesure 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 par 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. Des informations plus détaillées sont décrites en 4.5.2.

Compte tenu de ce comportement, un UIP doit être développé pour la version CLR 4.0 ou une version ultérieure. Lorsque les versions CLR ne sont pas compatibles, l'UIP doit être démarré dans un processus séparé. L'UIP n'est 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 les UIP sur différentes versions incompatibles du CLR.

Lorsqu'un Client FDI détecte qu'un UIP exige un CLR qui n'est pas compatible avec le Client FDI, le Client FDI peut utiliser une classe "proxy" qui permet une interaction avec l'UIP construit, au moyen d'une version différente du CLR.

Le Client FDI charge un exécutable de l'UIP du proxy, crée une instance de la classe "proxy" et délègue l'exécution de l'UIP à ce proxy. Le proxy démarre un processus avec le CLR exigé 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.

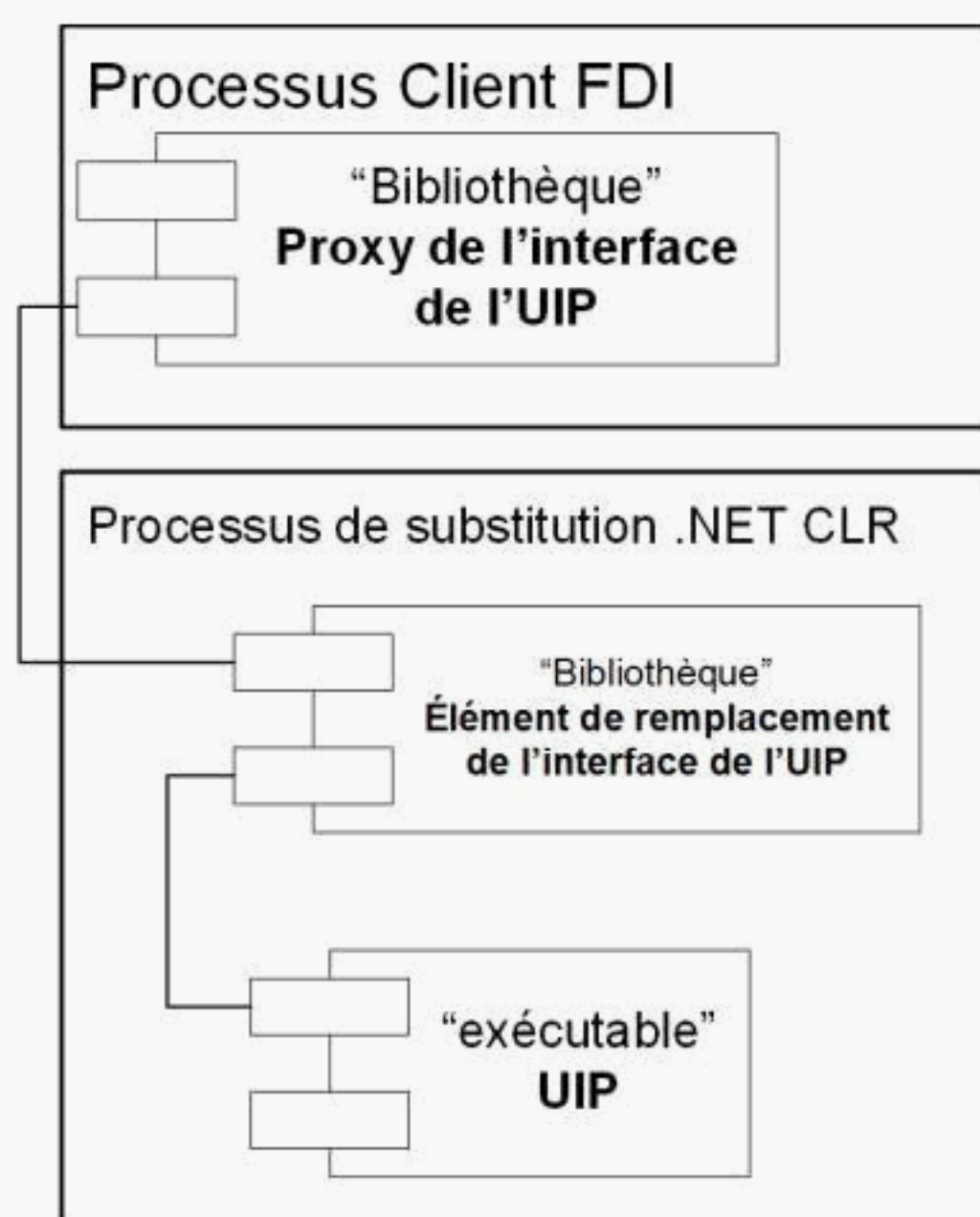


Figure 2 – Processus de substitution .NET

#### 4.5.3 Comment identifier la 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 la Figure 3).

```
clrVersion = Assembly.LoadFrom(<Chemin de l'assemblage>).ImageRuntimeVersion;
```

IEC

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 actuel. Le compilateur ne lit pas ce fichier.

#### 4.6 Déploiement de l'UIP

Les règles générales d'installation de l'UIP sont décrites dans l'IEC 62769-2. L'exécutable de l'UIP ne doit pas être enregistré dans le Cache global d'Assemblages.

La règle "nom fort" assure que les Assemblages connexes de différentes versions de l'UIP peuvent coexister pendant la durée d'exécution.

La mise en œuvre du Client FDI assure que le déploiement de l'UIP fonctionne indépendamment des informations d'identification 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" exigent des droits "Administrateur". Le système d'exploitation Windows fournit plusieurs moyens qui permettent à une application en cours d'exécution avec des droits utilisateur limités d'exécuter des actions avec des privilèges administrateur transparents pour 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. La solution alternative consiste à copier les exécutables de l'UIP dans des dossiers inscriptibles pour les utilisateurs "ordinaires".

<sup>1</sup> Visual Studio est un exemple de produit approprié disponible sur le marché. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit cité.



## 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 variations d'états sont décrits en 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 par l'appel des fonctions correspondantes 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 Étapes d'activation de l'Assemblage UIP

#### 4.7.2.1 Load (charger)

Le Client FDI doit charger les exécutables de l'UIP au moyen du 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.
- Lorsqu'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.
- Lorsqu'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 de fichier différent importe peu. Seule l'identité de l'Assemblage est importante.
- Lorsqu'un Assemblage est chargé avec LoadFrom, et le chemin de vérification inclut un Assemblage avec la même identité (par exemple, dans le Cache global d'Assemblages ou dans un répertoire de l'application), cet Assemblage est alors chargé, même si un chemin de fichier différent a été spécifié.
- LoadFrom exige les permissions `FileIOPermissionAccess.Read` et `FileIOPermissionAccess.PathDiscovery` ou `WebPermission`, sur le chemin spécifié.
- LoadFrom charge l'assemblage dans le Domaine d'Application par défaut.
- Lorsqu'une image native d'un Assemblage (générée par ngen.exe) existe pour le chemin de fichier spécifié, elle n'est alors pas utilisée. L'Assemblage ne peut pas être chargé comme indépendant du domaine, c'est-à-dire que 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 qui concernent 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.GetType` et

`System.Activator.CreateInstance`. La bibliothèque de types FDI déclare un "attribut personnalisé" dénommé `UIPActivationClass`. Cet attribut doit seulement être ajouté à l'objet qui met en œuvre l'interface `IDtmUiFunction` qui met réellement en œuvre la fonction de démarrage 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 cette utilisation, 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 exiger d'entrées dans le registre Windows ou dans le Cache global d'Assemblages;
- 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 qui contient la classe "main" de l'UIP de mise en œuvre de 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'utilisation de bibliothèques externes ne doit pas altérer 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 bibliothèques externes partagées**

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

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

Lorsqu'un Assemblage partagé est utilisé, alors les règles suivantes s'appliquent.

- Toute modification incompatible de l'Assemblage partagé doit conduire à une nouvelle identité, par exemple, différent numéro de version.
- Les Assemblages partagés ne doivent pas prétendre être chargés à partir d'un chemin d'installation spécifique, par exemple, compter sur le fait que certains fichiers sont stockés dans le même répertoire ou dans un sous-répertoire.
- Les variables statiques dans les Assemblages partagés sont également partagées lorsque 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 Paquetages FDI qui utilisent 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 pour ce qui concerne la structure de données internes. La logique du destructeur doit être limitée pour détruire l'objet pour ce qui concerne les 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 Étapes 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`. Par suite d'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 lorsque le `ApplicationDomain` lui-même est détruit. Cette situation signifie que si le Client FDI charge un Assemblage UIP dans le `ApplicationDomain` par défaut, ces Assemblages et tous les Assemblages dépendants ne sont alors jamais déchargés à moins que l'application soit fermée.

Les Assemblages UIP doivent être développés compte tenu de ce comportement .NET Framework. Pour réduire la consommation de mémoire, les règles suivantes s'appliquent.

- Réduire le plus possible 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. Ces éléments 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 dans les UIP qui proviennent de différents fournisseurs, un UIP peut déléguer au Client FDI la présentation et le traitement d'actions supplémentaires spécifiques à l'UIP. Néanmoins, les UIP sont autorisés à 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 varier 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 varie (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 présenter toutes les actions personnalisées exposées par un UIP avec des éléments dédiés de l'Interface Utilisateur.

### 4.8.2 Exécution d'un 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 (chronophage) de la fonction nommée `OperationName` doit être exécutée dans un fil différent. La synchronisation avec le fil appelant est traitée par 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` est terminé, l'attribut `IsCompleted` des objets `IAsyncResult` doit être défini sur `True`. Lorsque 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`. Lorsque `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`.

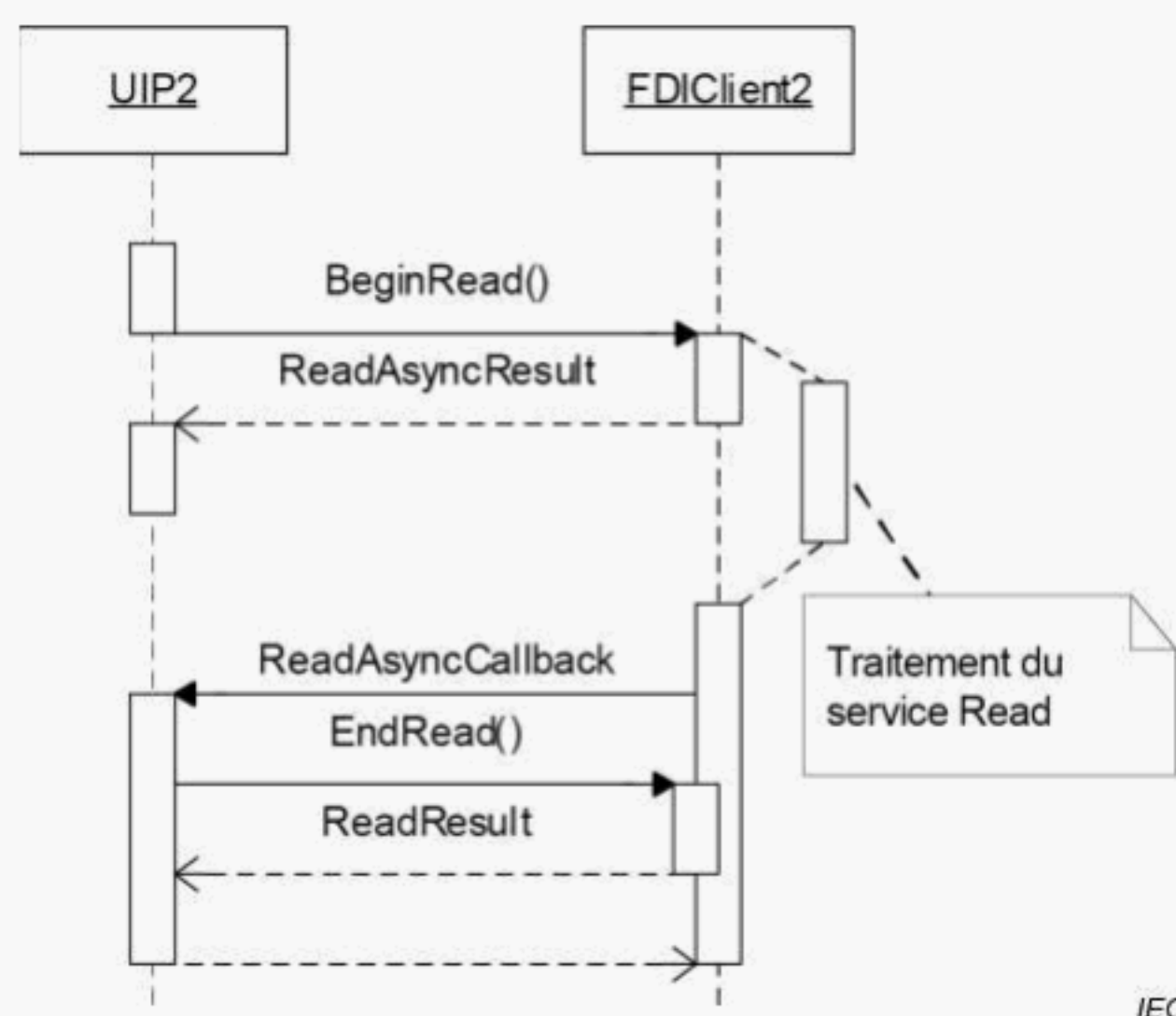
L'objet `IAsyncResult` retourné par `Begin OperationName` contient l'objet `WaitHandle`. L'UIP doit effectuer une synchronisation de son propre fil au moyen de l'objet `WaitHandle`.

#### 4.8.2.3 Séquence d'exécution d'un service sans blocage

Le paragraphe suivant spécifie 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 repose 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. *EndOperationName* 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 d'un service asynchrone fondé sur *IAsyncPattern*.

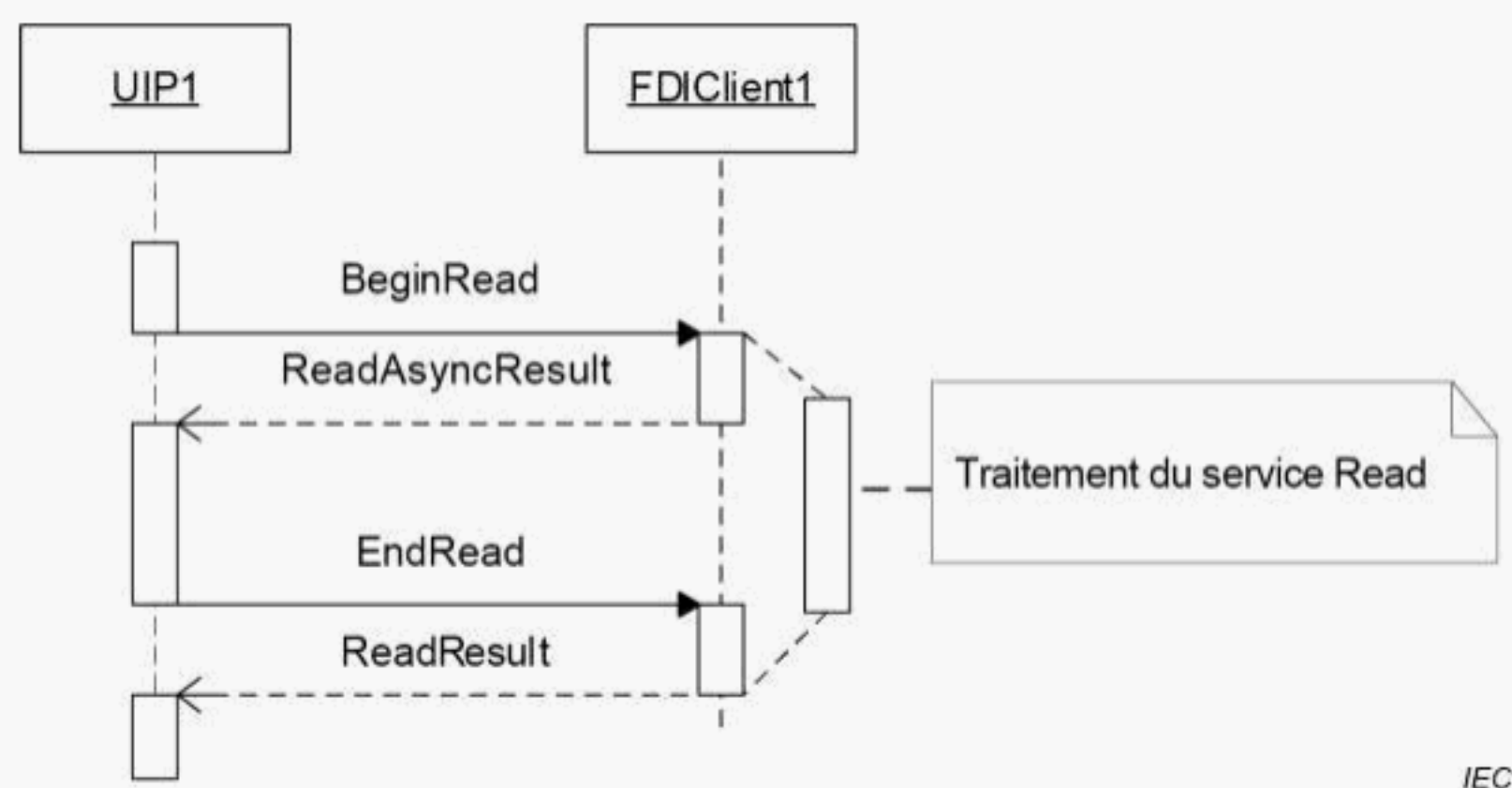


IEC

**Figure 4 – Exemple d'exécution d'un service asynchrone fondé sur *IAsyncPattern***

#### 4.8.3 Exécution d'un service de blocage

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



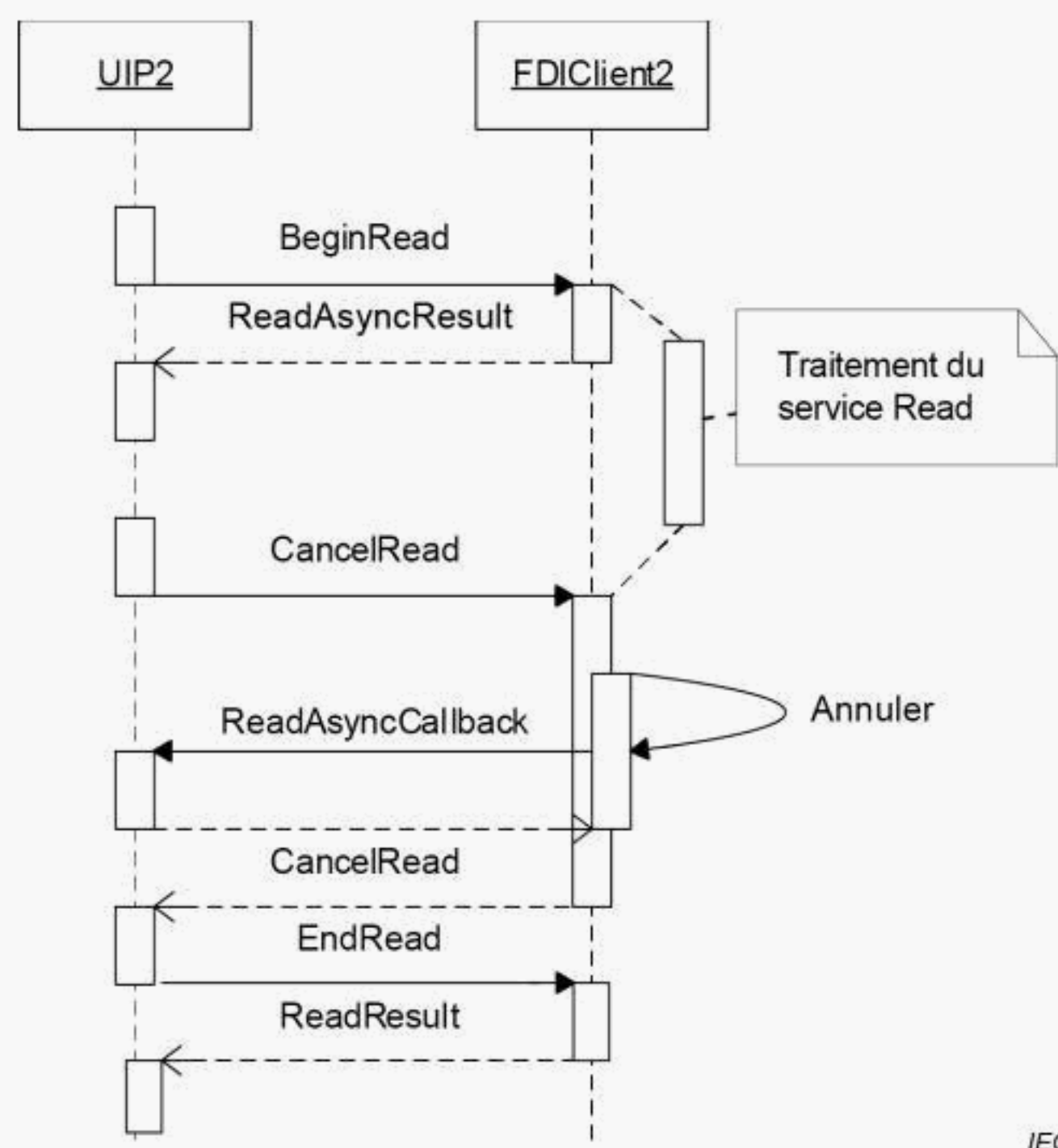
IEC

*ReadAsyncResult* est l'objet qui met en œuvre l'interface *IAsyncResult*.

**Figure 5 – Exemple d'exécution d'un service de blocage avec le modèle fondé sur *IAsyncResult***

#### 4.8.4 Exécution du service Cancel

Certains services spécifiés pour l'interface *IDeviceModel* (voir le 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".



IEC

**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 dès que possible. Après annulation de l'opération, le Client FDI notifie l'UIP à travers la `ReadAsyncCallback`. L'UIP doit alors appeler la fonction `EndRead`.

NOTE Un défi général qui met en œuvre ce modèle consiste à traiter des accrochages correctement sur les deux côtés (UIP2 et FDIClient2). Dans le cas où le Client FDI a transmis l'exécution du service par l'intermédiaire d'un service OPC UA, l'exécution réelle du service se déroule 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 dès que le Serveur FDI a terminé la demande de service. La réponse associée envoyée par le Serveur FDI peut être parvenue au Client FDI (ou pas). Le Client FDI peut invoquer la fonction `ReadAsyncCallback` tandis que l'UIP invoque la fonction `CancelRead`.

`ReadAsyncResult` est l'objet qui met 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 impasses et les boucles infinies.

L'UIP ne doit pas exécuter d'opérations synchrones comme décrit en 4.8.3 dans le fil de l'interface utilisateur: Le fil de l'interface utilisateur d'un processus doit être dédié à la réception des entrées de l'utilisateur et à l'exécution des tâches de dessin seulement.

L'UIP et le Client FDI ne doivent pas bloquer le fil de l'interface utilisateur. L'interface utilisateur doit toujours rester réactive. Le fil de l'interface utilisateur est partagé entre les différents objets liés à l'interface utilisateur FDI pour l'entrée utilisateur et les opérations de dessin. Lorsqu'un objet bloque ce fil afin d'effectuer un certain traitement, ce blocage affecte la réactivité d'autres interfaces utilisateur.

L'UIP et le Client FDI ne doivent pas bloquer un appel à la méthode `Begin OperationName` : une méthode `Begin OperationName` ne doit démarrer qu'une opération asynchrone. L'appelant ne doit pas être bloqué.

#### 4.8.6 Expiration de délai

Les interfaces référencées à l'Article 5 permettent l'exécution de services asynchrones. Le temps pour l'exécution de ces services dépend de contraintes de performance liées à la communication de bus et aux performances du Client FDI/Serveur FDI. Les règles énumérées ci-dessous visent l'interopérabilité du système en ce qui concerne la prévention des "Accrochages". La règle générale stipule qu'il est admis que le composant gère le traitement d'expiration de délai seulement pour les processus qui sont complètement sous le contrôle de ce composant. La liste suivante indique les éléments de l'ensemble du système qui sont autorisés à mettre en œuvre la fonction de détection d'expiration de délai.

- UIP: Il ne doit pas mettre en œuvre la détection d'expiration de délai.
- Logique Applicative: Elle ne doit pas mettre en œuvre la détection d'expiration de délai (Paquetage FDI).
- Client FDI: Il doit mettre en œuvre la détection d'expiration de délai. Dans le cas d'une architecture OPC UA, le support connexe est intégré dans les piles de communication OPC UA. L'expiration de délai détectée pendant les opérations effectuées pour le compte de l'UIP doit être transmise comme des codes de résultat négatif de la fonction.
- Serveur FDI: Il doit mettre en œuvre la détection d'expiration de délai. Dans le cas d'une architecture OPC UA, le support connexe est intégré dans les piles de communication OPC UA.
- Serveur de communication: Il met en œuvre la détection d'expiration de délai pour la connexion OPC UA conformément aux spécifications OPC UA. Le support connexe est intégré dans les piles de communication OPC UA. En outre, le Serveur de Communication met en œuvre une détection d'expiration de délai limitée au réseau directement connecté au port physique connecté au Serveur de Communication.

#### 4.8.7 Traitement des exceptions

Un objectif important de la spécification consiste à établir une distinction claire entre les problèmes de qualité logicielle et les états prévus de traitement. Par conséquent, la spécification définit deux catégories générales d'exceptions:

- a) Les exceptions qui indiquent des états ou des événements de logiciels qui n'ont pas été prévus lors du développement de logiciels sont considérées comme des problèmes de qualité logicielle (erreur d'exécution).
- b) Les exceptions qui indiquent des défaillances prévues d'exploitation de logiciels.

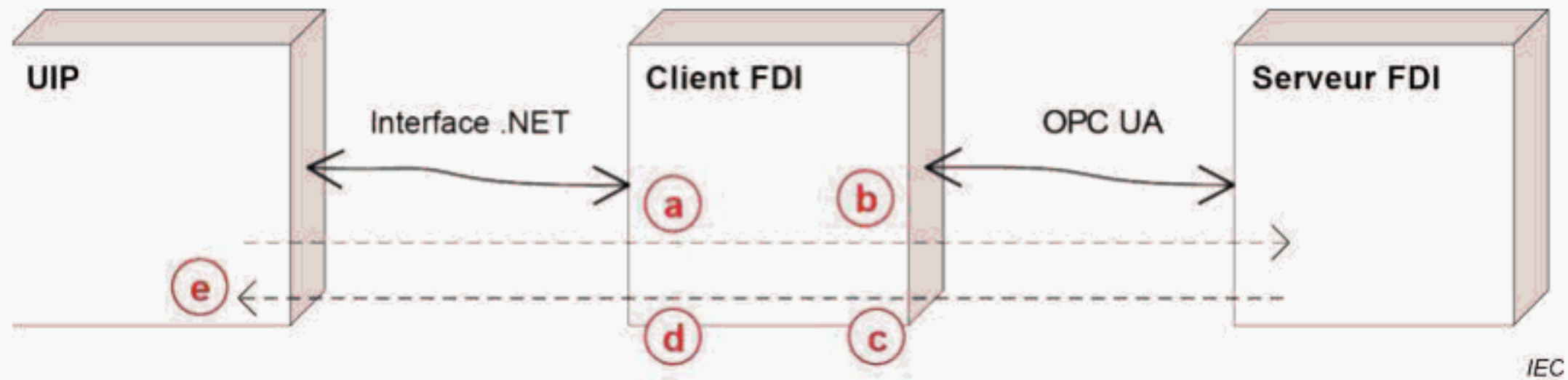
Exemples de problèmes de qualité logicielle, indiqués par les exceptions:

- discordance de type d'argument de fonction;
- discordance de plage de valeurs d'argument de fonction;
- division par zéro;
- référence au pointeur NULL.



Exemples pour le traitement anticipé des erreurs:

- gestion de problèmes de communication;
- traitement de données E/S générales;
- erreurs d'entrée utilisateur.



**Figure 7 – Source d'exception**

Selon l'architecture FDI, les exceptions peuvent se produire à différentes étapes du traitement de service (voir la Figure 7):

- transfert de la demande de l'UIP au Client FDI;
- transfert de la demande à l'intérieur du Client FDI;
- traitement de la réponse à partir du Serveur FDI;
- transfert de la réponse à l'UIP;
- traitement de la réponse à l'intérieur de l'UIP.

Les problèmes de traitement de services détectés à l'intérieur du Serveur FDI et au-delà sont traités par les résultats des services définis par l'architecture OPC UA.

En ce qui concerne la mise en œuvre du modèle `IAsyncResult`, les règles suivantes s'appliquent.

- Tout échec à l'étape a) doit être indiqué par une exception générée par `Begin OperationName`.
- Tout échec pendant les étapes b) à e) doit être traité par le composant correspondant. L'exécution d'`End OperationName` doit alors indiquer l'échec par une exception.

#### 4.8.8 Interfaces de type sûr (type safe)

Le Modèle d'Information héberge des variables d'appareils de différents types. Les valeurs de ces variables sont transférées à l'aide de la classe `DataValue` (FDI Interfaces and Data Types.chm).

Les Services d'Accès à l'Appareil prennent en charge l'écriture ou la lecture de plusieurs variables dans un seul service. Le type de données choisi pour le transport de données est `DataValue` qui met en œuvre le transport de type sûr à cause de la propriété `Datatype` de `DataValue` qui décrit la valeur "type de données" au moyen de l'énumération `Datatype`. Puisque les fonctions `get/set` de la propriété `Value` de `DataValue` utilisent le type de données "Object" pour transférer la valeur réelle, le récepteur de données (UIP) doit vérifier le type de données avant le traitement des données.

#### 4.8.9 Globalisation et localisation

La prise en charge de la localisation de l'UIP peut être mise en œuvre par le biais de fichiers ressources (`.res(x)`) ou par le biais d'Assemblages satellites.

Le Client FDI doit configurer les paramètres régionaux et les informations relatives aux pays qui doivent être utilisés par l'UIP au moyen des arguments `currentRegion` et `currentCulture` qui sont soumis avec l'invocation de la méthode `Fdi.Dtm.Ui.IDtmUiFunction.Init`. L'UIP ne doit pas dériver les informations relatives au paramètre régional au moyen de la fonction `Thread.CurrentUICulture`. Le type de données pour `currentRegion`



est `RegionInfo` défini dans l'espace de noms `.NET System.Globalization`. Le type de données pour `currentCulture` est `CultureInfo` défini dans l'espace de noms `.NET System.Globalization`.

#### 4.8.10 Traitement de contrôle WPF

Lorsque la mise en œuvre d'un UIP se fonde sur `UserControl` WPF, l'UIP hérite de l'interface de la classe `UserControl`, ce qui signifie qu'il y a plus d'attributs et d'événements de méthodes disponibles pour le Client FDI qui ne sont pas couverts par la spécification FDI. Inversement, un UIP met en œuvre la fonction d'accessibilité. D'une part, les règles liées affectent la qualité du produit UIP; d'autre part, elles affectent l'interopérabilité.

#### 4.8.11 Traitement des formes de Windows

Lorsqu'une mise en œuvre de l'UIP se fonde sur les formes de Windows, l'UIP hérite de la classe `UserControl`, ce qui signifie qu'il y a plus d'attributs et d'événements de méthodes disponibles pour le Client FDI qui ne sont pas couverts par la spécification FDI. Inversement, un UIP met en œuvre la fonction d'accessibilité. D'une part, les règles liées affectent la qualité du produit UIP; d'autre part, elles affectent l'interopérabilité. Le Client FDI doit faire des appels à fil sécurisé aux commandes `Windows.Forms`.

### 4.9 Sécurité

#### 4.9.1 Généralités

L'objectif de la sécurité est de protéger un système contre les menaces qui compromettent la stabilité du système, l'intégrité et les données sensibles.

La sécurité de l'ensemble du système commence avec le processus de conception, qui ne relève pas du domaine d'application de la normalisation. Du point de vue du système, la sécurité consiste à contrôler l'accès aux ressources, telles que les composants d'applications, les données et le matériel. Le `.NET Framework` fournit un support pour contraindre l'accès aux ressources. La sécurité du système repose sur le contrôle des permissions d'accès.

Une approche différente repose sur la certification et l'authentification. Étant donné qu'il est nécessaire de soumettre tous les Paquetages FDI à des essais de conformité et de certification, il est admis par hypothèse que ces Paquetages FDI certifiés ne représentent aucune menace pour un système. Cette disposition signifie qu'un UIP peut être exécuté avec des permissions de confiance totale.

Alors qu'un système soumis à des contraintes excessives peut conduire à des problèmes fonctionnels, des permissions non contraintes peuvent être perçues comme des menaces de sécurité. Le paragraphe 4.9 représente un compromis entre les deux méthodes.

#### 4.9.2 Permissions d'accès

##### 4.9.2.1 Généralités

Le présent paragraphe spécifie les permissions et restrictions spécifiques à la technologie en plus de celles spécifiées dans l'IEC 62769-2. Les permissions et restrictions doivent être appliquées par le Client FDI. Les règles de mise en œuvre définissent la manière dont ces paramètres doivent être mis en œuvre.

##### 4.9.2.2 Permissions et restrictions de l'UIP spécifiques à la technologie

Les permissions et restrictions générales de l'UIP sont spécifiées dans l'IEC 62769-2. Les permissions et restrictions spécifiées en 4.9.2 sont spécifiques aux UIP fondés sur `.NET`.

- a) Le lancement d'un composant Active-X n'est pas permis.
- b) Un UIP ne doit pas écrire sur le registre du système d'exploitation. L'accès en lecture au registre est admis.

Le Client FDI doit limiter les permissions de l'UIP selon cette liste.

#### 4.9.2.3 Règles de mise en œuvre

Afin de limiter les permissions d'un UIP, un Client FDI doit exécuter un UIP dans un processus séparé (pas le processus du Client FDI) qu'il convient d'exécuter avec un compte utilisateur séparé (4.5.2 décrit la manière dont un Client FDI peut gérer l'hébergement d'un UIP dans un processus différent). Par la limitation des permissions du compte utilisateur séparé, le Client FDI limite également les permissions de l'UIP en cours d'exécution avec le compte utilisateur séparé.

La méthode de création et de configuration du compte utilisateur séparé est spécifique à l'hôte. Elle

- indique s'il s'agit d'un compte local ou d'un compte de domaine;
- indique si le compte utilisateur séparé est utilisé uniquement pour l'hébergement de l'UIP ou également à d'autres fins;
- indique si le compte utilisateur séparé est réutilisé pour tous les processus de l'UIP de la même instance de Client FDI ou de plusieurs Clients FDI;
- indique comment les restrictions sont mises en œuvre (par exemple, accès limité au système de fichiers par le mécanisme du système d'exploitation, blocage de l'accès au réseau par un pare-feu).

Il dépend uniquement du Client FDI d'exécuter chaque instance de l'UIP dans son propre processus ou d'exécuter plusieurs ou toutes les instances de l'UIP dans le même processus.

Étant donné que les UIP s'exécutent dans un processus séparé, il est nécessaire que les développeurs d'UIP sachent que l'ouverture d'un dialogue modal avec les mécanismes du système d'exploitation assure uniquement la modalité dans Windows pour le processus concerné. Pour éviter ce comportement, les développeurs d'UIP peuvent démarrer l'UIP modal ou démarrer un nouvel UIP modal.

#### 4.9.3 Concept d'identité de code

La capacité à identifier de manière unique les exécutables de l'UIP contribue à la sécurité du système.

Comme il a été précédemment décrit en 4.3, les exécutables de l'UIP doivent être signés avec des noms forts. Les Assemblages .NET signés avec des noms forts permettent:

- l'identification unique des exécutables de l'UIP;
- la vérification d'intégrité de codes.

NOTE L'avantage des exécutables de l'UIP avec des noms forts est perdu si cet Assemblage charge dynamiquement d'autres Assemblages de bibliothèques qui ne sont pas signés avec des noms forts.

### 5 Définition d'interface

Les tableaux suivants (Tableau 2 au Tableau 9) spécifient le mapping entre les services abstraits spécifiés dans l'IEC 62769-2 et la mise en œuvre .NET correspondante qui se trouve dans le fichier de bibliothèque de types "FDI.DLL" et un fichier d'aide lié "FDI Interfaces and Data Types.chm" (Interfaces et Types de données FDI.chm).

NOTE Les fichiers "FDI.DLL" et "FDI Interfaces and Data Types.chm" peuvent être obtenus auprès des organisations de bus de terrain. (Consulter aussi le site web <http://www.fdi-cooperation.com>)

Le Tableau 2 spécifie le mapping des Services de Propriété de Base.

**Tableau 2 – Services de Propriété de Base**

Service abstrait	Mise en œuvre .NET
GetDeviceAccessInterfaceVersion	IDeviceAccess.Version
GetOnlineAccessAvailability	IDeviceAccess.OnlineAccessAvailable

Le Tableau 3 spécifie le mapping des Services de Modèle d'Appareil.

**Tableau 3 – Services de Modèle d'Appareil**

Service abstrait	Mise en œuvre .NET
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

Le Tableau 4 spécifie le mapping des Services de Contrôle d'Accès.

**Tableau 4 – Services de Contrôle d'Accès**

Service abstrait	Mise en œuvre .NET
InitLock	IAccessControl.BeginInitLock IAccessControl.EndInitLock
ExitLock	IAccessControl.BeginExitLock IAccessControl.EndExitLock

Le Tableau 5 spécifie le mapping des Services d'Accès Direct.

**Tableau 5 – Services d'Accès Direct**

Service abstrait	Mise en œuvre .NET
InitDirectAccess	IDirectAccess.BeginInitDirectAccess IDirectAccess.EndInitDirectAccess
ExitDirectAccess	IDirectAccess.BeginExitDirectAccess IDirectAccess.EndExitDirectAccess
Transfer	IDirectAccess.BeginTransfer IDirectAccess.EndTransfer

Le Tableau 6 spécifie le mapping des Services d'Hébergement.

**Tableau 6 – Services d'Hébergement**

Service abstrait	Mise en œuvre .NET
GetClientTechnologyVersion	Fdi.Frame.IFrame.Version
OpenUserInterface a)	Fdi.Frame.Ui.IFrameUi.BeginOpenDtmUiModal Fdi.Frame.Ui.IFrameUi.EndOpenDtmUiModal Fdi.Frame.Ui.IFrameUi.BeginOpenDtmUi Fdi.Frame.Ui.IFrameUi.EndOpenDtmUi
CloseUserInterface a)	Fdi.Dtm.Ui.CloseMeRequestHandler b) Fdi.Frame.Ui.IFrameUi.BeginCloseDtmUi Fdi.Frame.Ui.IFrameUi.EndCloseDtmUi
LogAuditTrailMessage	Fdi.Frame.IAuditTrail.Notify
SaveUserSettings	Fdi.Frame.IUserSettings.SaveUserSettings
LoadUserSettings	Fdi.Frame.IUserSettings.LoadUserSettings
Trace	Fdi.Frame.ITrace.TraceData Fdi.Frame.ITrace.TraceEvent
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
ExportFile	Fdi.Frame.ImportExport.IFileExport.BeginInitExport Fdi.Frame.ImportExport.IFileExport.EndInitExport Fdi.Frame.ImportExport.IFileExport.BeginWriteExport Fdi.Frame.ImportExport.IFileExport.EndWriteExport Fdi.Frame.ImportExport.IFileExport.BeginFinishExport Fdi.Frame.ImportExport.IFileExport.EndFinishExport
CancelExportFile	Fdi.Frame.ImportExport.IFileExport.BeginFinishExport Fdi.Frame.ImportExport.IFileExport.EndFinishExport
ImportFile	Fdi.Frame.ImportExport.IFileExport.BeginInitImport Fdi.Frame.ImportExport.IFileExport.EndInitImport Fdi.Frame.ImportExport.IFileExport.BeginReadImport Fdi.Frame.ImportExport.IFileExport.EndReadImport Fdi.Frame.ImportExport.IFileExport.BeginFinishImport Fdi.Frame.ImportExport.IFileExport.EndFinishImport
CancellImportFile	Fdi.Frame.ImportExport.IFileExport.BeginFinishImport Fdi.Frame.ImportExport.IFileExport.EndFinishImport
OpenDefaultApplication	Fdi.Frame.DefaultApplication.IOpenDefaultApplication.BeginInitOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.EndInitOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.BeginWriteOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.EndWriteOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.BeginFinishOpenDefaultApplication Fdi.Frame.DefaultApplication.IOpenDefaultApplication.EndFinishOpenDefaultApplication



Service abstrait	Mise en œuvre .NET
GetEnvironmentProperties	Fdi.Frame.HostProperty. GetHostingProperties
a) Les fonctions <code>OpenUserInterface</code> , <code>CloseUserInterface</code> doivent être lancées uniquement avec le modèle de fonctionnement décrit en 4.8.2.3. b) À utiliser par l'UIP pour qu'il s'arrête de lui-même.	

Le Tableau 7 spécifie le mapping des Services d'UIP.

**Tableau 7 – Services d'UIP**

Service abstrait	Mise en œuvre .NET
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	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) La réponse spécifiée <code>deactivateCancelled</code> du service <code>Deactivate</code> est mappée à l'exception <code>FdiCannotCloseUiException</code> à générer par l'UIP lorsque l'UIP rencontre des problèmes avec la désactivation.	

Le Tableau 8 spécifie le mapping des types de données de base.

**Tableau 8 – Types de données de base**

Type de données de base	Mise en œuvre .NET	
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

Le Tableau 9 spécifie le mapping des types particuliers de données.

**Tableau 9 – Types particuliers**

Type particulier de données	Mise en œuvre .NET	
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	

Les matrices de données peuvent être transportées au moyen de la classe `Fdi.DataTypes.ArrayValue`.

Une définition et une documentation détaillées de l'interface sont disponibles dans les fichiers suivants:

- FDI.DLL (Assemblage .NET)
- FDI Interfaces and Data Types.chm (Fichier d'Aide)







