IEC 62769-6

Edition 3.0 2023-04
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

Field device integration (FDI®) – Part 6: FDI® Technology Mappings

IEC 62769-6:2023-04 RLV(en)

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

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

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

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

IEC publications search - 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 It also gives information on projects, replaced and withdrawn publications.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
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
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.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
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 62769-6

Edition 3.0 2023-04
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

Field device integration (FDI®) – Part 6: FDI® Technology Mappings

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 25.040.40; 35.100.05

ISBN 978-2-8322-6837-7

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI®) –

Part 6: FDI® Technology Mappings

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.

This redline version of the official IEC Standard allows the user to identify the changes made to the previous edition IEC 62769-6:2021. A vertical bar appears in the margin wherever a change has been made. Additions are in green text, deletions are in strikethrough red text.

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. It is an International Standard.

This third edition cancels and replaces the second edition published in 2021. This edition constitutes a technical revision.

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

a) Separated each technology mapping out to subparts of Part 6 (i.e., Part 6-xxx)

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

Draft	Report on voting
65E/867/CDV	65E/924/RVC

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

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 document 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: FDI® Technology Mappings

1 Scope

This part of IEC 62769 specifies the technology mapping for the concepts described in the Field Device Integration (FDI®[1]) standard. The technology mapping focuses on implementation of the components FDI® Client and User Interface Plug-in (UIP) ~~that are specific only to~~ in the specified technologies for the WORKSTATION platform~~/.NET~~ and the MOBILE platform as defined in IEC 62769-4. There are individual subparts for the currently supported technologies .NET and HTML5.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. 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 62769-6-100, *Field Device Integration (FDI®) – Part 6-100: Technology Mapping – .NET*

IEC 62769-6-200, *Field Device Integration (FDI®) – Part 6-200: Technology Mapping – HTML5*

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

FCG TS10099, *Field Device Integration (FDI®) – Technology Management*

HTML5, W3C Recommendation. World Wide Web Consortium (W3C) (2014)

~~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*~~

[1] FDI® is a registered trademark of the non-profit organization Fieldbus Foundation, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

ECMA-262, *ECMAScript 2016 Language Specification*

3 Terms, definitions, abbreviated terms, ~~symbols~~ acronyms and conventions

3.1 Terms and definitions

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

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

- IEC Electropedia: available at https://www.electropedia.org/
- ISO Online browsing platform: available at https://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 and acronyms

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

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

~~**3.3 Symbols**~~

~~Figures in this document use 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)

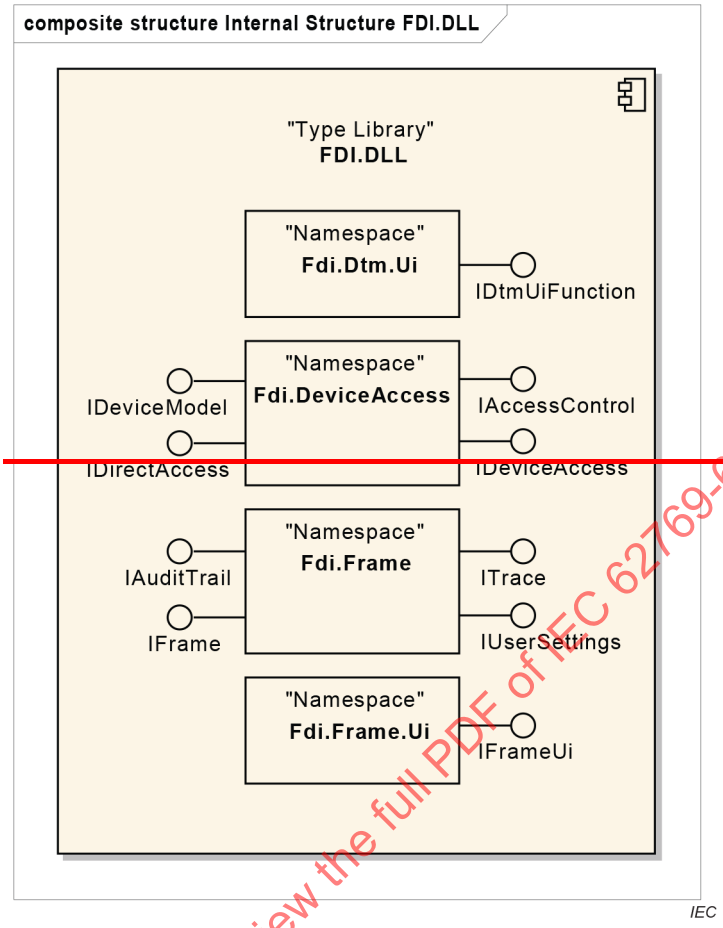
4.1.3 FDI Type Library

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.)

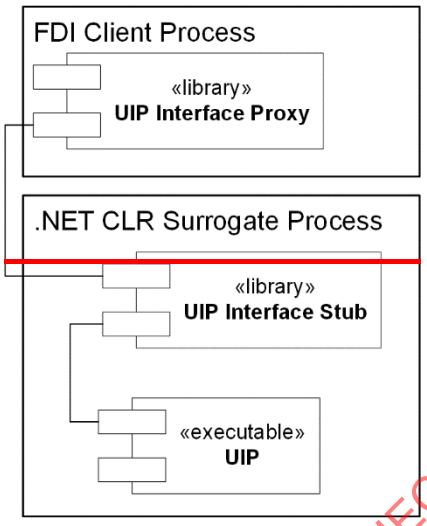
Taking this behavior in account, a UIP shall be developed for CLR version 4.0 or any future version. If the CLR versions do not match, the UIP shall be started in a separate process. The UIP will then not be displayed as an integrated module within the FDI Client. It is up to the FDI Client to realize 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[2] 2008 and 2010 IDE allow developers to select the .NET Framework target. The selection of a .NET Framework target older than the base for the current Visual Studio IDE automatically creates a configuration file listed as "app.config" within the solution explorer. This file only reflects the current complier setting. The compiler does not read that file.

4.6 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.

[2] 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 `IAsyncResult` object.

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

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

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

4.8.2.2 UIP internal functions

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

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

4.8.2.3 Non-blocking service execution sequence

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

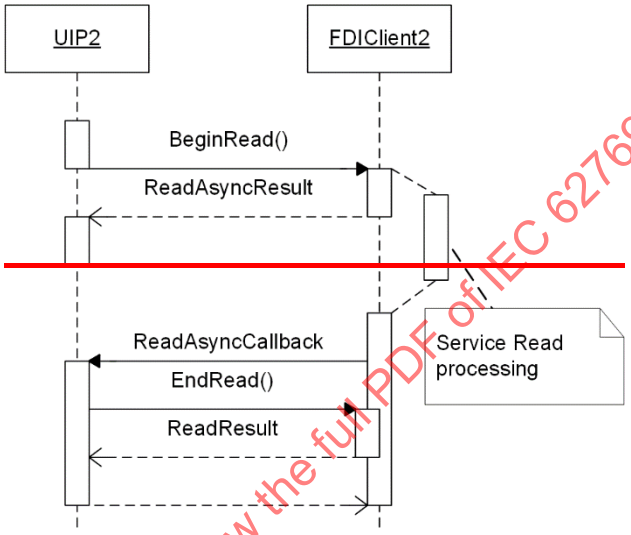
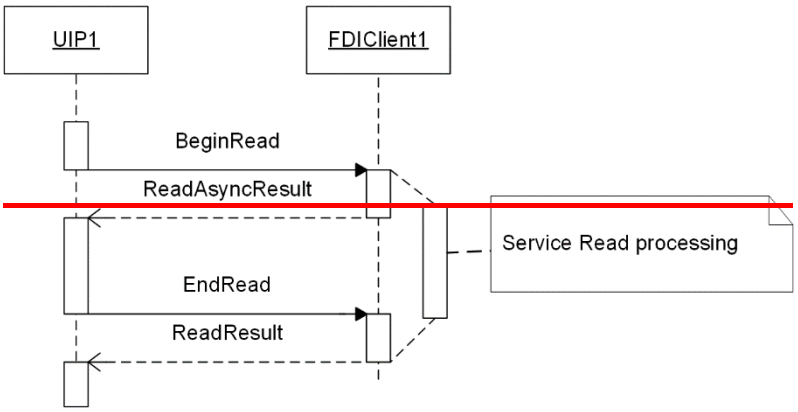


Figure 4 – IAsyncPattern based asynchronous service execution example

4.8.3 Blocking service execution

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



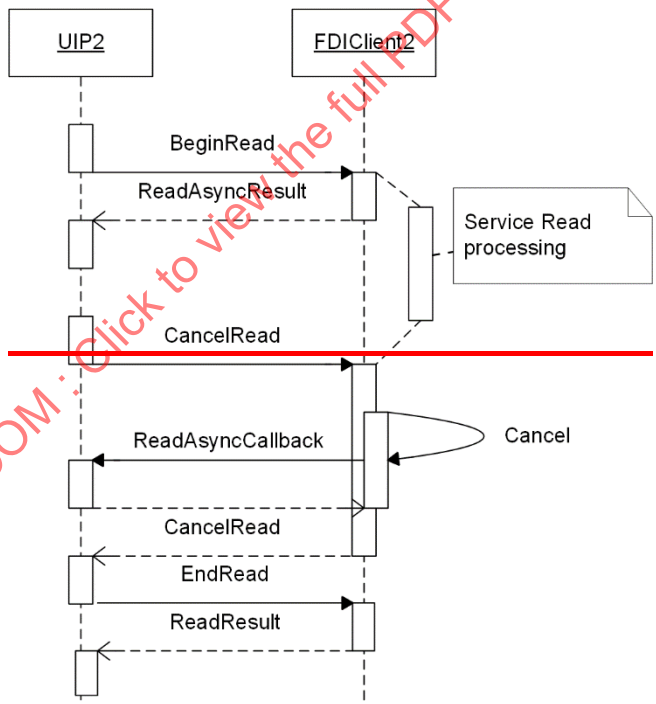
IEC

`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 `IAsyncResult`.

4.8.5 Threading

4.8.5.1 Implementation rules

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

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

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

The UIP shall not run synchronous operations as described in 4.8.3 in the user interface thread: the user interface thread of a process shall be dedicated to 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:

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

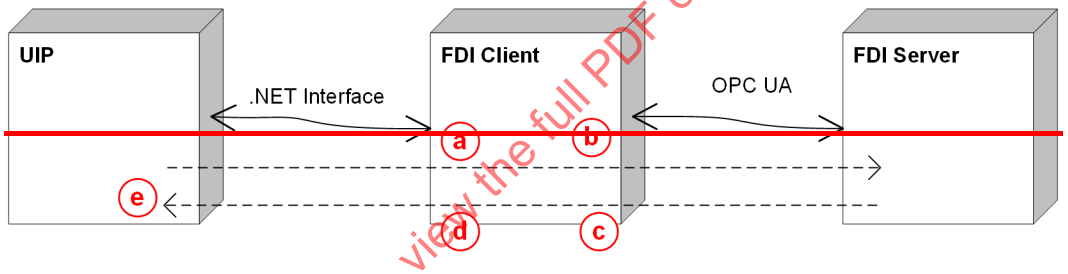
b) Exceptions indicating anticipated software operation failures.

Examples of software quality issues indicated by exceptions are:

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

Examples of anticipated error handling are:

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



IEC

Figure 7 – Exception source

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

a) passing the request from the UIP to the FDI Client;

b) request forwarding inside the FDI Client;

c) processing the response from the FDI Server;

d) forwarding the response to the UIP;

e) response processing inside the UIP.

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

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

– Any failure occurring with step a) shall be reported by an exception thrown by the Begin*OperationName*.

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

4.8.8 Type safe interfaces

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

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

4.8.9 Globalization and localization

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[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

Abstract Service	.NET Implementation
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
CancelImportFile	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
ApplicationSpecificActionItemsChangeCallback	Fdi.Dtm.Ui. IApplicationSpecificActions.ApplicationSpecificActionItemSetChanged
a) The Deactivate service specified response `deactivateCancelled` maps to the exception `FdiCannotCloseUiException` to be thrown by the UIP if the UIP has problems with the deactivation.	

Table 8 specifies the mapping of the base data types.

Table 8 – Base Data Types

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

Table 9 specifies the mapping of the special data types.

Table 9 – Special Types

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

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

Detailed interface definition and interface documentation are available in:

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

4 Platforms and runtime

4.1 Runtime

4.1.1 .NET Framework

A UIP for the .NET-based runtime shall be built upon the Microsoft.NET™[3] Framework and executed in the .NET Common Language Runtime. The RuntimeId is ".NET Framework CLR<<CLR_Version>>". The concrete versions of the required .NET Framework and the CLR are specified in FCG TS10099 FDI® Technology Management.

IEC 62769-6-100 specifies the detailed technology mapping.

4.1.2 HTML5

A UIP for the HTML5-based runtime shall be built upon HTML5 (see HTML5, W3C Recommendation), ECMAScript®[4] (see ECMA-262) and Cascading Style Sheets. The RuntimeId is "FDIHtml<<Version>>". The version and thus the concrete features to be supported are specified in FCG TS10099 FDI® Technology Management.

IEC 62769-6-200 specifies the detailed technology mapping.

4.2 Platforms

4.2.1 Workstation

The individual RuntimeIds to be supported by an FDI® Client for the Workstation platform are specified in FCG TS10099.

The PlatformId for the Workstation platform is “Workstation”.

4.2.2 Mobile

The RuntimeIds to be supported by an FDI® Client for the Mobile platform are specified in FCG TS10099.

The PlatformId for the Mobile platform is “Mobile”.

---

3 Microsoft.NET™ is the is the trademark of a product supplied by Microsoft Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

4 ECMAScript® is a registered trademark of a product supplied by Ecma International. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Bibliography

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

IEC 62769-6

Edition 3.0 2023-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

Field device integration (FDI®) – Part 6: FDI® Technology Mappings

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

IEC 62769-6:2023-04(en-fr)

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI®) –

Part 6: FDI® Technology Mappings

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.

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. It is an International Standard.

This third edition cancels and replaces the second edition published in 2021. This edition constitutes a technical revision.

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

a) Separated each technology mapping out to subparts of Part 6 (i.e., Part 6-xxx)

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

Draft	Report on voting
65E/867/CDV	65E/924/RVC

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

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.

FIELD DEVICE INTEGRATION (FDI®) –

Part 6: FDI® Technology Mappings

1 Scope

This part of IEC 62769 specifies the technology mapping for the concepts described in the Field Device Integration (FDI®[1]) standard. The technology mapping focuses on implementation of the components FDI® Client and User Interface Plug-in (UIP) in the specified technologies for the WORKSTATION platform and the MOBILE platform as defined in IEC 62769-4. There are individual subparts for the currently supported technologies .NET and HTML5.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

IEC 62769-6-100, *Field Device Integration (FDI®) – Part 6-100: Technology Mapping – .NET*

IEC 62769-6-200, *Field Device Integration (FDI®) – Part 6-200: Technology Mapping – HTML5*

FCG TS10099, *Field Device Integration (FDI®) – Technology Management*

HTML5, W3C Recommendation, World Wide Web Consortium (W3C) (2014)

ECMA-262, *ECMAScript 2016 Language Specification*

3 Terms, definitions, abbreviated terms, acronyms and conventions

3.1 Terms and definitions

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

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

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

[1] FDI® is a registered trademark of the non-profit organization Fieldbus Foundation, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

3.2 Abbreviated terms and acronyms

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

CLR	Common Language Run-time
UML	Unified Modeling Language

4 Platforms and runtime

4.1 Runtime

4.1.1 .NET Framework

A UIP for the .NET-based runtime shall be built upon the Microsoft.NET™[2] Framework and executed in the .NET Common Language Runtime. The RuntimeId is ".NET Framework CLR<<CLR_Version>>". The concrete versions of the required .NET Framework and the CLR are specified in FCG TS10099 FDI® Technology Management.

IEC 62769-6-100 specifies the detailed technology mapping.

4.1.2 HTML5

A UIP for the HTML5-based runtime shall be built upon HTML5 (see HTML5, W3C Recommendation), ECMAScript®[3] (see ECMA-262) and Cascading Style Sheets. The RuntimeId is "FDIHtml<<Version>>". The version and thus the concrete features to be supported are specified in FCG TS10099 FDI® Technology Management.

IEC 62769-6-200 specifies the detailed technology mapping.

4.2 Platforms

4.2.1 Workstation

The individual RuntimeIds to be supported by an FDI® Client for the Workstation platform are specified in FCG TS10099.

The PlatformId for the Workstation platform is “Workstation”.

4.2.2 Mobile

The RuntimeIds to be supported by an FDI® Client for the Mobile platform are specified in FCG TS10099.

The PlatformId for the Mobile platform is “Mobile”.

---

2 Microsoft.NET™ is the is the trademark of a product supplied by Microsoft Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

3 ECMAScript® is a registered trademark of a product supplied by Ecma International. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.