IEC 62769-6-200

Edition 1.0 2023-04

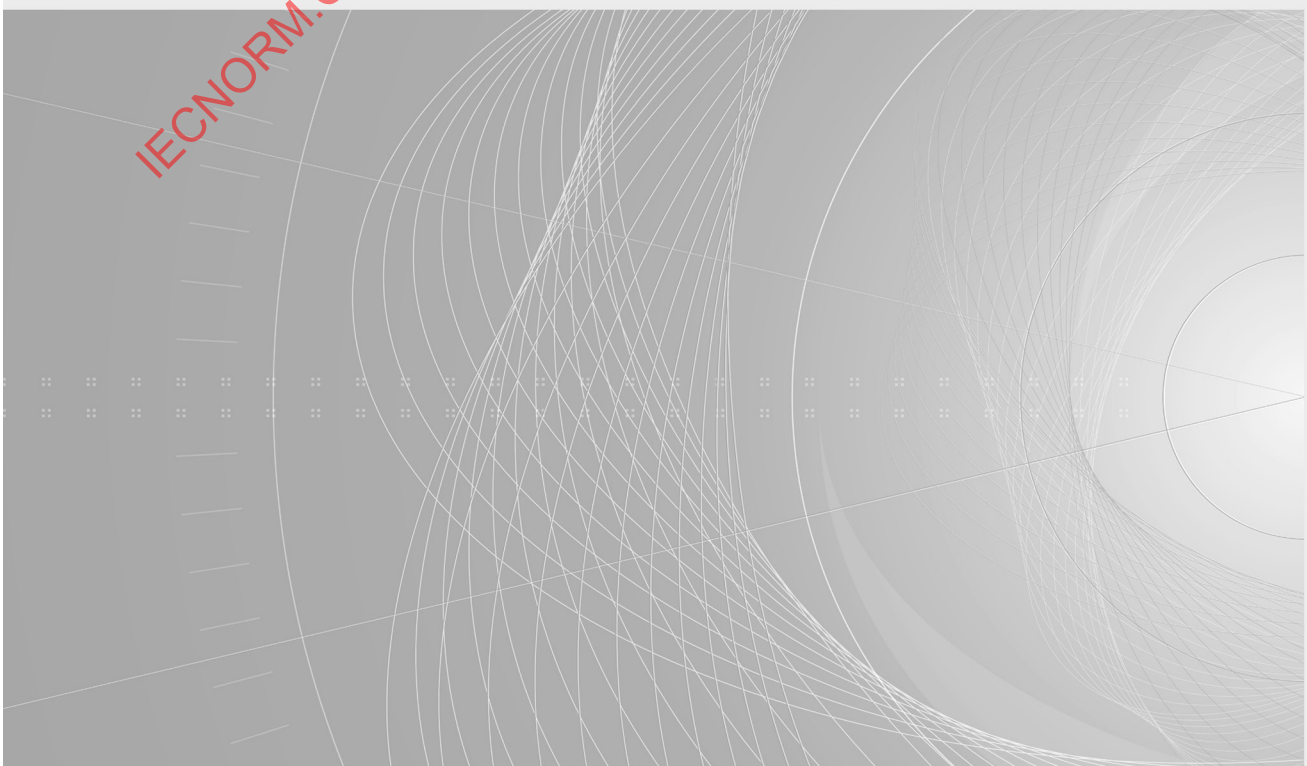
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

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

Intégration des appareils de terrain (FDI®) – Partie 6-200: Mapping de technologies – HTML5



IEC 62769-6-200:2023-04(en-fr)

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 It also gives information on projects, replaced and withdrawn publications.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.

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

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

Service Clients - 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.

IEC Products & Services Portal - products.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
Le premier dictionnaire d'électrotechnologie en ligne au monde, avec plus de 22 300 articles terminologiques en anglais et en français, ainsi que les termes équivalents dans 19 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

IEC 62769-6-200

Edition 1.0 2023-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

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

Intégration des appareils de terrain (FDI®) – Partie 6-200: Mapping de technologies – HTML5

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040.40; 35.100.05

ISBN 978-2-8322-6820-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI®) –

Part 6-200: Technology Mapping – HTML5

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 c0 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.

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

Draft	Report on voting
65E/870/CDV	65E/926/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/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 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.

FIELD DEVICE INTEGRATION (FDI®) –

Part 6-200: Technology Mapping – HTML5

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 regarding the components FDI® Client and User Interface Plug-in (UIP) for the Runtime 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.

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

IEC 62769-1:2021, *Field device integration (FDI®) – Part 1: Overview*

IEC 62769-2:2021, *Field device integration (FDI®) – Part 2: 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*

W3C HTML5.0, *W3C Recommendation HTML5 – A vocabulary and associated APIs for HTML and XHTML*

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, IEC 62769-6-100, as well as the following apply.

ISO and IEC maintain terminology 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

[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.

3.1.1
HTML5 runtime
functional component of the FDI® Client, which executes the HTML5-based UIP. In principle, it provides the functionality of a web browser (HTML rendering, JavaScript execution engine)

3.1.2
FDI® Host Type Library
host specific implementation of FDI® Type Library which consists of host.js and fdi.js. FDI® Host Type Library is provided by the host vendors and will be deployed together with the UIP on UIP startup

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.

UML Unified Modeling Language

3.3 Conventions

Figures in this document use the graphical symbols according to ISO/IEC 19505-1 (UML 2.0).

4 Technical concepts

4.1 General

4.1.1 Overview

In 4.1.2, this document describes the technology base for UIP implementation based on HTML5, the software environment including the related implementation rules. Clause 4 follows a life-cycle (use case) oriented approach.

Subclause 4.3.4 describes the copy deployment procedures and related implementation rules for the UIP and the FDI® Client. UIP instantiation and termination is described in 4.5. Subclause 4.6 defines the rules about interaction between the FDI® Client and the UIP. Security related definitions are written in 4.7. The service interface definitions for the FDI® Client and the UIP are found in Clause 5.

4.1.2 FDI® Type Library

The Device Access Services, the Hosting Services, and the UIP Services are modelled as TypeScript interfaces passing TypeScript 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 TypeScript error objects are defined.

The FDI® TypeScript interfaces, data types, and exception classes are defined in a single FDI® Type Library. The file name of the interface definition shall be ‘fdi.ts’.

Figure 1 shows the FDI® Type Library structure.

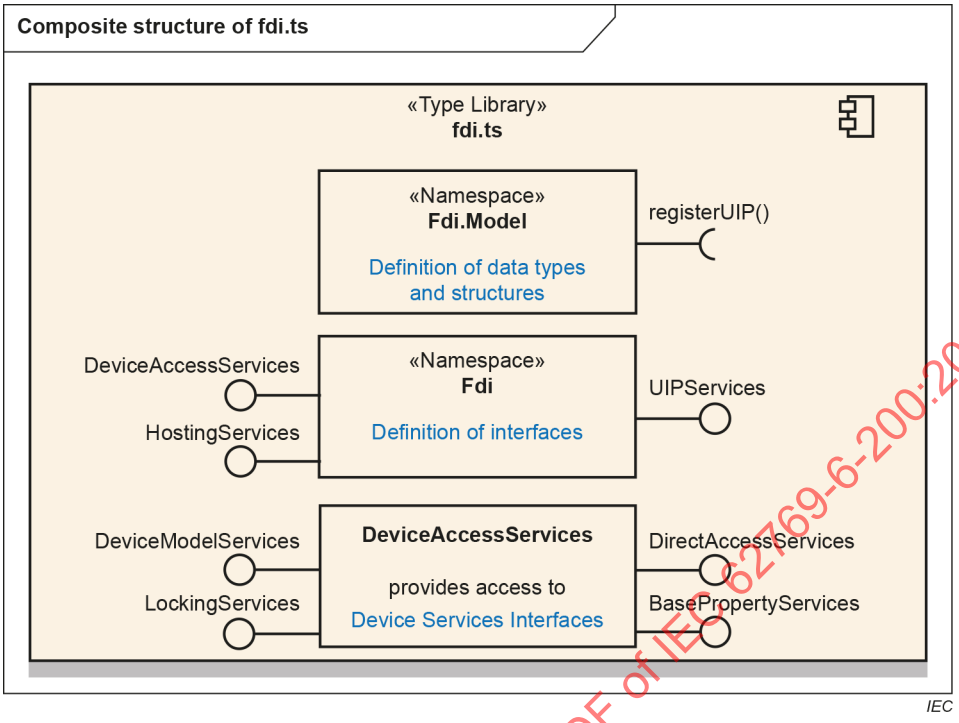


Figure 1 – FDI® Type Library structure

NOTE 1 The composite structure diagram shows only the core interfaces that implement the interfaces defined in IEC 62769-2.

The interfaces shall be implemented by the FDI® Client respectively the UIP using TypeScript. The TypeScript implementation files are transpiled to JavaScript using the feature set specified in the edition of ECMA 262, which corresponds to the RuntimeId of the UIP variant. The edition of ECMA 262 is unambiguously specified in the FCG TS10099 by the RuntimId of the UIP variant.

The result of the transpilation of FDI®.ts is FDI®.js, which contains the function prototypes. The transpiled FDI® Type Library, i.e. FDI®.js, and the host specific implementation of the interfaces, i.e. host.js, is called FDI® Host Type Library. The overall structure of FDI® Type Library together with host and UIP specific implementations of the interfaces is shown in Figure 2.

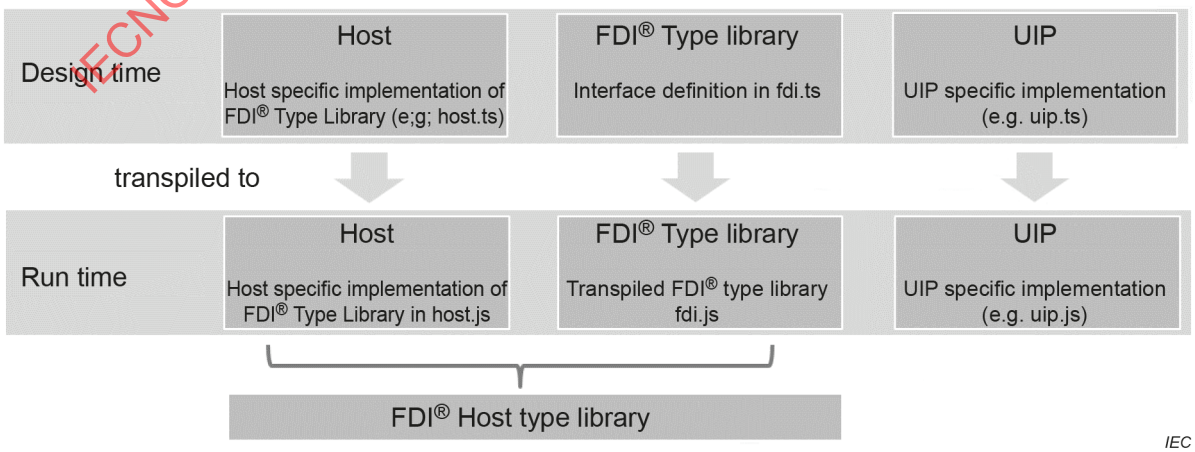


Figure 2 – Relationship between FDI® Type Library, host specific and UIP specific implementation of the interfaces

NOTE 2 The FDI® Type Library is specified by this specification and can be obtained from FieldcommGroup (see www.fieldcommgroup.org). Host specific implementation will be provided by host vendors and the UIP specific implementation will be part of the specific UIP variant.

The FDI® Type Library fdi.ts respectively the fdi.js shall be versioned as per IEC 62769-1:2021, 8.1. The FDI® Type Library is part of the FDI® Core Technology as per IEC 62769-1:2021, 8.3.2.1. and therefore directly influences the FDI® Technology Version. All Compatible changes of the fdi.ts respectively fdi.js 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 62769-1:2021, 8.3.2.2).

The FDI® Type Library shall be installed as part of every FDI® Client installation. User Interface Plug-Ins (UIP) and the FDI® Client Application shall use this instance of the fdi.js. 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.

The transpiled host specific implementation of the FDI® Type Library, i.e. host.js, and the FDI® Type Library itself (fdi.js) shall be provided by the FDI® Client during the load process of the UIP. Both files shall be located in the subfolder ‘./scripts’ of the UIP installation folder. The respective files of the FDI® Host Type Library fdi.js and host.js shall be referenced by the UIP explicitly. For example, this can be done with the statements <script src="./scripts/ fdi.js" type="module"/> and <script src="./scripts/host.js" type="module"/> within the <header>-statement of the HTML file named according to the <StartElementName>-Attribute of the HTML5 UIP Package.

4.2 UIP representation

4.2.1 Self containment

The UIP Variant can contain either a single or multiple HTML files and their related supplementary files (for example resource files). The main entry HTML file is located directly in the UIP installation directory. Its name is given by the <StartElementName>-Attribute of the HTML5 UIP Package.

Besides the FDI® Host Type Library, i.e. FDI®.js and host.js, the UIP Variant shall be self-contained. All supplementary files required by the UIP are stored under (a) subfolder(s) of the installation directory. The FDI® Host Type Library is provided by the FDI® Client.

Figure 3 shows an example of a UIP with subfolders for scripts, Cascading Style Sheets and images.

```
async function browseParameterList() {

    var node = new UIPTypes.NodeSpecifier("parameterSet", false);                    // create NodeSpecifier to be browsed
    var cancelToken = new UIPTypes.CancelToken(guid());                              // create cancel token to enable cancelling of call
    tokenDirectory[cancelToken.id] = cancelToken;

    let fdiDMS = deviceModelServices;                                                // interface to host specific implementation of device model services

    await fdiDMS.browse(node,cancelToken).then(                                      // async call to browse function
        response => { BrowseResponse(response, "Browse"); },                         // response evaluation, if call returns successful
        error => {                                                                   // response evaluation, if call returns wit failure
            if(error == Fdi.Model.StatusCode.Bad_RequestCancelled){                  // call was cancelled
                log("async call to Browse with ID:" + cancelToken.id + " cancelled");
                delete tokenDirectory[cancelToken.id];
            }
            else{                                                                    // other error inside browse function call
                log("async call to Browse with ID:" + cancelToken.id + " failed");
                delete tokenDirectory[cancelToken.id];
            }
        }
    ).catch(                                                                         // call to browse service failed
        exception => { log("async call to Browse with ID:" + cancelToken.id + " catched");
                delete tokenDirectory[cancelToken.id];
        }
    );
};
```

IEC

Figure 3 – Examples of UIP supplementary data files include javascript files, css files, and images

4.2.2 External libraries and supplementary files

The UIP can depend on external libraries (third party libraries) and other components, for example, specific user control libraries, icons, or images. FDI® Clients dynamically load and execute the UIP in the HTML5 runtime. To support this usage, as well as the requirement to prevent possible problems of conflicting libraries, the following rules are specified for external libraries and supplementary files.

External libraries and supplementary files shall:

- be contained within the UIP variant, except of the FDI® Host Type LIbrary (FDI®.js and host.js), which are provided by the FDI® Client;
- not require any installation procedure other than copying into the UIP installation directory or its subfolders;
- adhere to the access restrictions described in 4.7.2;
- be compatible with the platforms and runtimes the UIP Variant is built for.

The FDI® Client loads the HTML5 UIP by navigating to the main entry file given by the <StartElementName>-Attribute of the HTML5 UIP Package. Referenced external libraries and supplementary files have to be loaded explicitly by the UIP itself (e.g. stated as <script src="./scripts/myExternalLibrary.js" />). All references to external libraries and supplementary files within the UIP shall be specified as relative URLs.

4.3 UIP compatibility rules

4.3.1 Overview

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

An FDI® Client shall provide the versions of the FDI® Host Type Library (fdi.js andhost.js) and the HTML5 runtime supporting the feature set, which correspond to the RuntimeId of the UIP variant.

4.3.2 FDI® Type Library Compatibility Strategy

The UIP shall check for the FDI® Technology Version of the FDI® Client using the HostingService GetClientTechnologyVersion. The following rules ensure the compatibility of the UIP with the FDI® Host Type Library:

- If the UIP variant was built for a completely compatible FDI® Technology Version (same major and minor version), the UIP can be directly started in the FDI® Client.
- If the minor version of the FDI® Technology of the FDI® Client is lower than the minor version of FDI® Technology the UIP has been built for, the UIP may only use the interface methods defined in the FDI® Technology Version supported by the FDI® Client.
- If the UIP variant to be started was built for an incompatible FDI® Technology Version (different major versions), the FDI® Client shall start the UIP using the matching FDI® Host Type Library and Runtime.

4.3.3 Runtime compatibility strategy

In different runtime versions, UIPs may use a different, potentially incompatible feature set. The feature set, which shall be supported by any runtime version identified by a specific RuntimeId is specified in the FCG TS10099.

FDI® Clients supporting a specific RuntimeId shall support at least the features listed in the FCG TS10099 for the individual RuntimeId.

UIP variants built for a specific RuntimeId can rely on the features specified in the FCG TS10099 for this RuntimeId. If a UIP uses features, which are not listed in the standard feature set of the corresponding RuntimeId, the UIP shall check for the availability of the feature in the respective FDI® Client and implement a fallback strategy as for example an information message to the user.

If an FDI® Client detects that a UIP variant is built for a RuntimeId requiring another feature set from the runtime, the FDI® Client can use the appropriate runtime, e.g. the HTML5+ runtime, to execute the UIP (see Figure 4).

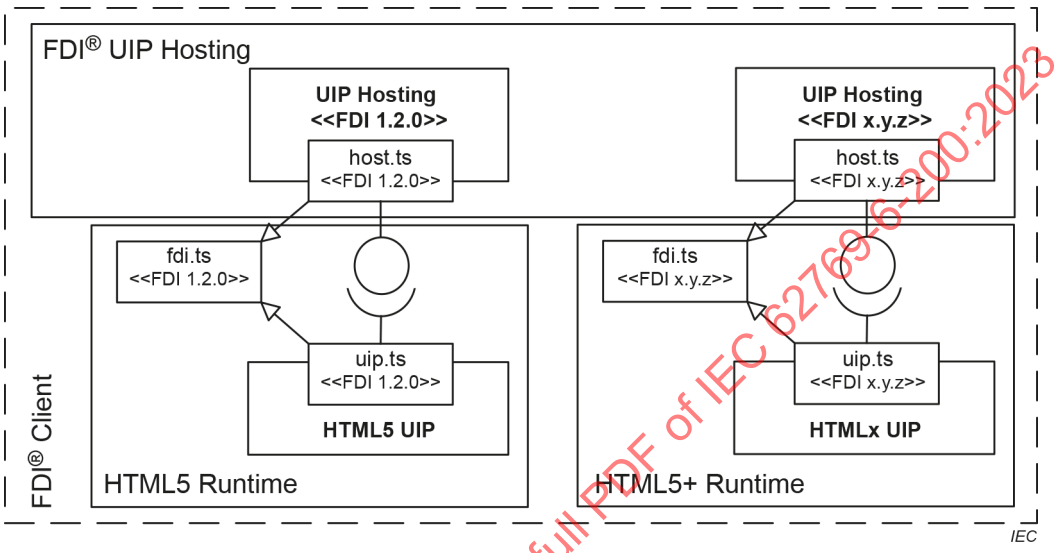


Figure 4 – Compatibility strategy to host UIPs built for different Runtime Versions

4.3.4 UIP executable compatibility rules

A HTML5 based User Interface Plugin is not compiled to machine code. Therefore, the element "CpuInformation" in the catalog.xml file, which holds the compilation target platform information, shall not be used for HTML5 based User Interface Plugins.

4.4 UIP Deployment

The general UIP installation rules are outlined in IEC 62769-2. If the UIP is stored on the local file system, the FDI® Client and the FDI® Server shall ensure the integrity of the UIP.

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

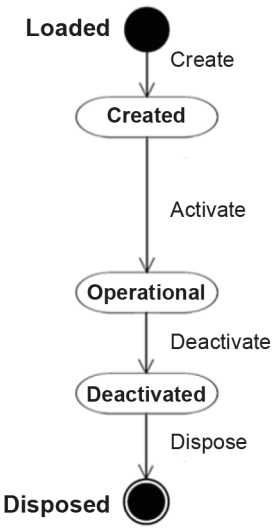
NOTE Certain operating system managed folders require specific access rights, for example, modifications in folder "Program Files" require "Administrator" rights. The Windows operating system provides several means to allow an application running with restricted user rights, to execute actions with administrator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

4.5 UIP Life-cycle

4.5.1 General

The UIP state machine, outlined in IEC 62769-2, is composed of the Loaded, Created, Operational, Deactivated and Disposed states. The mechanisms affecting state changes are described in 4.5.

After UIP deployment (see 4.3.4), the FDI® Client loads the HTML5 runtime dynamically into the memory and executes the related logic of the UIP by calling the corresponding FDI® specified interface functions. Figure 5 gives an overview on the UIP life-cycle.



Transition step	Source step	Destination step	Description
Create	Loaded	Created	The FDI® Client loads its HTML5 runtime, creates a browsing context for the UIP and navigates this browsing context to the main HTML entry page given by the <StartElementName>-Attribute of the HTML5 UIP Package. All resources referenced in the main HTML entry page shall be loaded as specified in W3C HTML5.
Activate	Created	Operational	The HTML5 Runtime sends the Javascript event "onload" to the UIP. The UIP shall bind its activation method to this event. Within this method it implements its startup and initialization sequence, which includes the registration of UIPServices interface implementation on the FDI® Client by calling Fdi.Model.registerUIP(). Finally, the FDI® Client calls UIPServices.activate() and the UIP reaches the state "Operational"
Deactivate	Operational	Deactivated	The FDI® Client shall call the UIPServices.deactivate() method by using the registered UIPServices interface. The UIP shall finish its operations and store data. The UIP shall bring itself into a state where it can be safely released.
Dispose	Deactivated	Disposed	The FDI® Client shall delete all references to the UIP instance.

IEC

Figure 5 – UIP Life-cycle

Subclauses 4.5.2 and 4.5.3 specify the rules about how the FDI® Client shall activate and deactivate the UIP.

4.5.2 UIP activation steps

4.5.2.1 Load

The FDI® Client shall load its HTML5 runtime. Within the HTML5 runtime the creation of the UIP shall be done by creating a browsing context for the UIP and navigating this browsing context to the main HTML entry page given by the `<StartElementName>`-Attribute of the HTML5 UIP Package. The `<StartElementName>`-Attribute is contained in the file uipcatalog.xml of the UIP Package.

4.5.2.2 Create

The UIP is created within the HTML5 runtime. All resources referenced in the main HTML entry page shall be loaded as specified in W3C HTML5.0.

When the UIP is created successfully, the HTML5 runtime shall send the Javascript event "onload" to the UIP.

4.5.2.3 Activate

The UIP shall bind its startup method to the Javascript event "onload". Within this method, the UIP implements its startup and initialization sequence.

Within the startup method the UIP shall call the method `Fdi.Model.registerUIP()` to enable the FDI® Client to call the methods on the UIP as specified in IEC 62769-2:2021, 6.1 UIP Services.

After registration of the UIP Services interface implementation of the specific UIP, the FDI® Client shall call the method `Fdi.UIPSerice.setSystemLabel()` to specify a human identifier of the UIP instance in the context of the FDI® Client. To finish UIP activation, the FDI® Client calls `Fdi.UIPSerice.activate()`. The invocation of `Fdi.UIPSerice.activate()` includes information on localization (RegionInfo, CultureInfo) and the calling context including the FDI® client specific implementation of the interfaces DeviceAccessServices and HostingServices. With successful completion of `Fdi.UIPSerice.activate()` the activation process of the UIP is finished and the UIP is operational.

The whole activation process is shown in Figure 6.

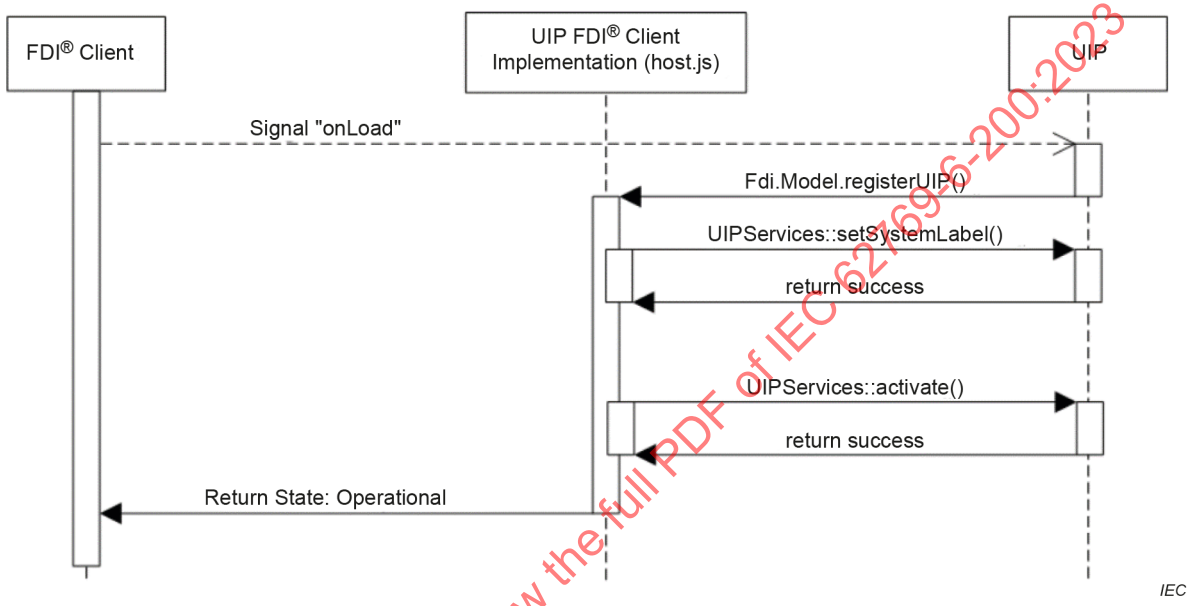


Figure 6 – UIP activation process

4.5.3 UIP deactivation

For UIP deactivation the FDI® Client shall call the method `Fdi.UIPService.deactivate()` using the implementation of the UIP Services interface registered during the activation process.

Afterwards the FDI® Client shall delete all references to the UIP instance.

4.5.4 General rules for UIP activation and deactivation

The activation logic shall be limited to instantiate the object in terms of the internal data structure and load of all referenced libraries. The deactivation implementation shall be limited to destroy the object in terms of releasing memory resources. The activation and the deactivation shall not:

- throw errors,
- invoke any call-back to the FDI® Client; and
- invoke any user interaction.

4.6 Interaction between an FDI® Client and a UIP

4.6.1 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
- specific UI Actions (UI actions, which are specific to one UIP, e.g. Reset, Reconnect).

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 methods `Fdi.HostingServices.StandardActionItemSetChanged()` and `Fdi.HostingServices..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.

4.6.2 Non-blocking service execution

4.6.2.1 FDI® Client internal functions

The productive (time consuming) part of a service named *OperationName* shall be performed asynchronously by the FDI® Client. For the asynchronous service execution, the async/await pattern based on Promises is used. Therefore, any method of the FDI® interface implementation returns a Promise (see Figure 7).

If the request cannot be executed, the FDI® Client shall release the Promise with "reject" to indicate that the requested operation didn't start. Any other response shall release the Promise with "resolve". The return value provided with the "resolve" shall include an appropriate error message and one of the status codes defined with `Fdi.Model.StatusCode`, if the time consuming operation resulted in an error or has been cancelled.

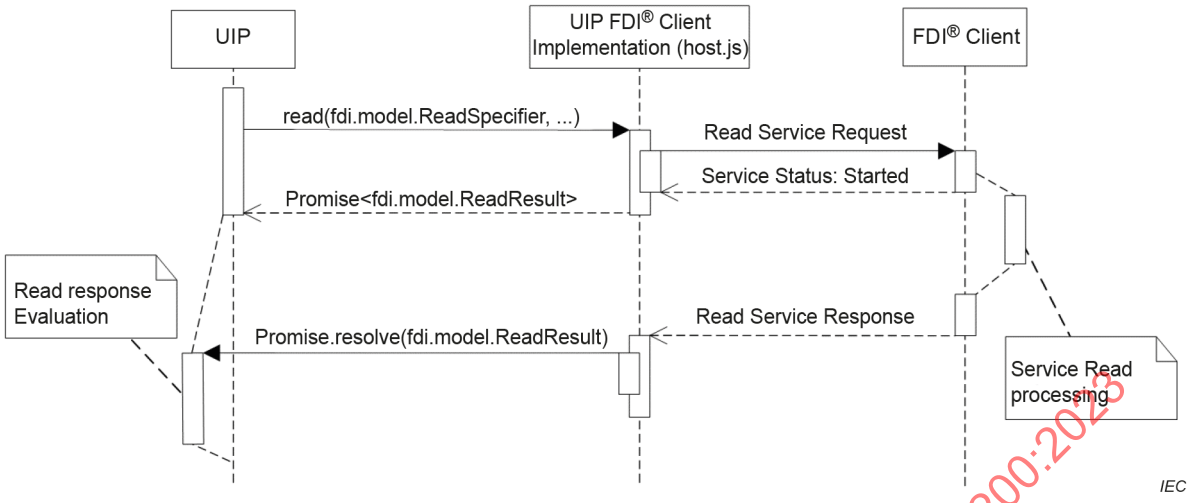


Figure 7 – Asynchronous service execution example (FDI® Client)

The implementation of Cancel*OperationName* is not implemented as separate service call. Instead, service calls that can be cancelled have an argument `cancelToken` of type `Fdi.Model.CancelToken`. If the UIP cancels the service execution, it calls the method `cancelToken.cancel()` which shall invoke the resolve of the Promise with `Fdi.Model.StatusCode.Bad_RequestCancelled` and an appropriate error message (see Figure 8).

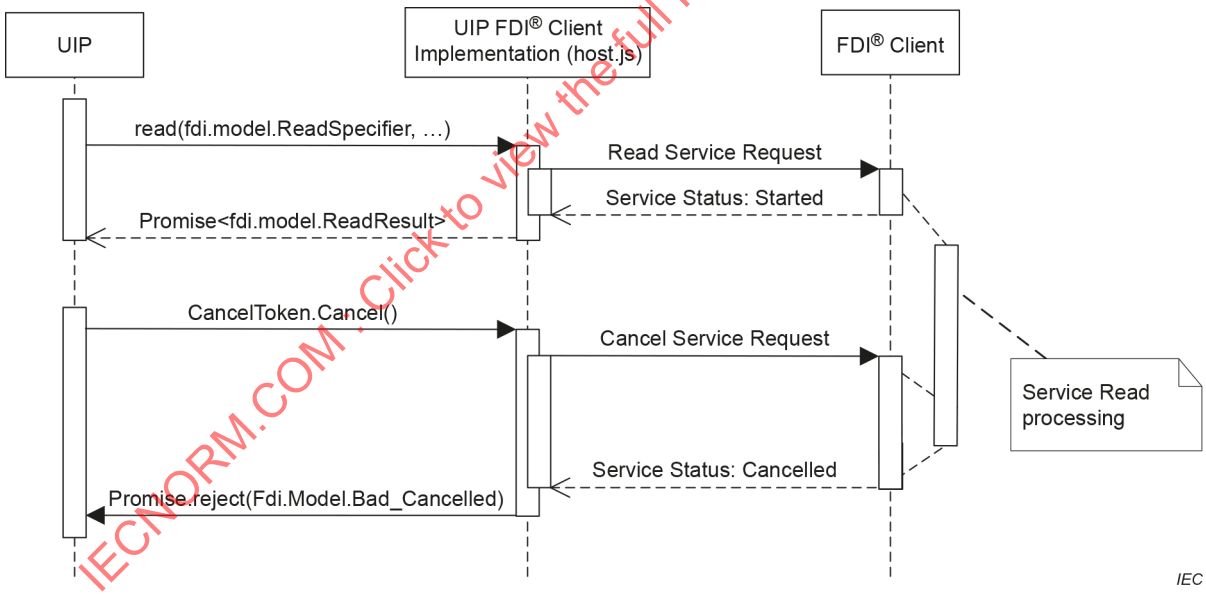


Figure 8 – Cancelled asynchronous service execution example (FDI® Client)

The handling of the operation within the UIP is described in the following 4.6.2.2.

4.6.2.2 UIP internal functions

The implementation of every service interface function that is called from the UIP, returns an asynchronous object of class Promise<OperationResult>". The UIP evaluates the Promise by implementing its ".then"-handler. In addition the ".catch()"-Handler should be implemented to catch unexpected errors. See example illustrated in Figure 9.

Each ".then"-handler shall have two async functions, one for "resolve" and one for "reject". An operation, which results in "resolve", has been executed successfully. An operation, which results in "reject", has either been cancelled or the service execution in the FDI® Client has resulted in an error. The ".catch"-handler additionally handles exceptions that may be thrown within the host specific implementation (e.g. failed type conversions, missing arguments).

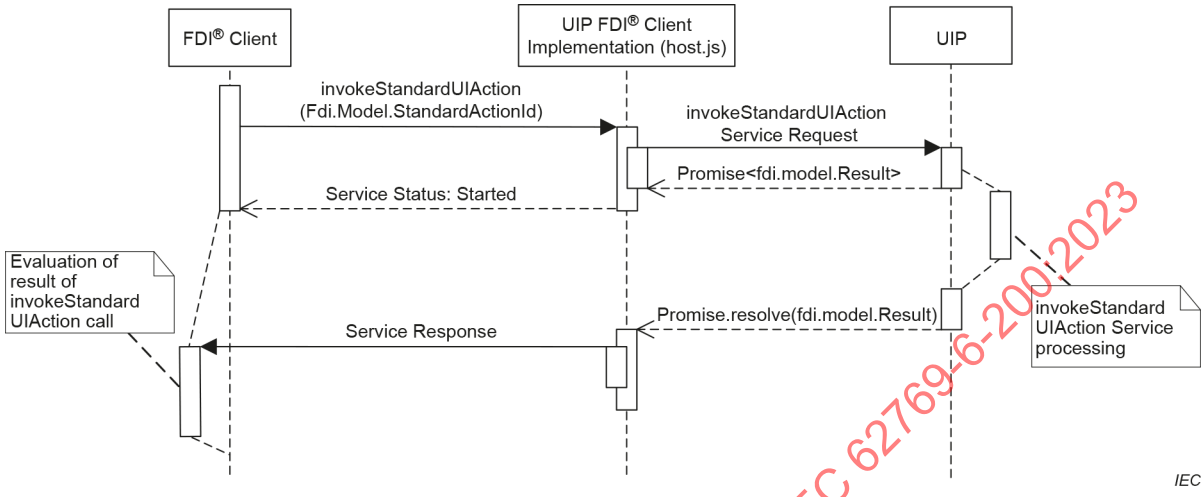


Figure 9 – Asynchronous service execution example (UIP)

The use of Promise implies, that the respective interface function is called from a function declared with the "async" keyword as shown in the code example in Figure 10. The "async" keyword enables asynchronous handling of the function. The interface function itself shall be called with the "await" statement. All implementation inside the ".then()"- or ".catch()"-handler shall be executed in a separate thread by the HTML5 engine and thus shall not block the UI.

```
async function OnBrowseButton() {                                  // event handler using 'async' keyword

  var fdiDMS = new DeviceModelServices();                          // interface pointer to Device Model Services
  var node = new Fdi.Model.NodeSpecifier("/ParameterSet/Tag", true);  // definition of node to browse

  var cancelTokenId = guid();
  var cancelToken = new Fdi.Model.CancelToken(cancelTokenId);      // creation of a cancel token

  await fdiDMS.browse(node, cancelToken)                           // browse service call
    .then(
      response => { EvalBrowseResponse(response, "Browse"); },     // first .then handler to cover 'resolve' of service
      error => {                                                   // second .then handler to cover 'reject' of service
        if (error.state == Fdi.Model.StatusCode.Bad_RequestCancelled) // handling of cancelled services
          log("async call to Browse cancelled.");
        else                                                       // common failure handling
          log("async call to Browse returned with failure.");

  }).catch(                                                        // service start fails, service not executed
    exception => {
      log("async call to Browse not executed.");
  });
};
```

IEC

Figure 10 – Simplified code sample to illustrate the described function implementation

4.6.3 Threading

4.6.3.1 Implementation rules

The UIP and FDI® Client shall not block the user interface thread. The user interface shall always stay responsive.

The FDI® Client, in particular the implementation provided with host.js, shall not block the UIP on method calls on the interfaces `Fdi.BasePropertyServices`, `Fdi.DeviceModelServices`, `Fdi.DirectAccessSerives`, `Fdi.LockingServices` and `Fdi.HostingServices`. The FDI® Client shall only start an asynchronous operation and return a *Promise* object for evaluation of asynchronous method execution result. The caller shall not be blocked.

The UIP shall not block the FDI® Client on method calls on the UIPServices interface. The UIP shall only start an asynchronous operation and return a *Promise* object for the evaluation of asynchronous method execution result. The caller shall not be blocked.

4.6.4 Timeout

The interfaces specified in Clause 5 enable asynchronous service execution. The time for the execution of such services depends on performance constraints related to bus communication or 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.
- FDI® Client: The FDI® Client shall implement timeout detection. In case of OPC UA, the related support is built into the OPC UA communication stacks. Timeout detected during operations performed on behalf of the UIP shall be forwarded as negative function result codes.

4.6.5 Exception handling

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

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

Examples of software quality issues indicated by exceptions are:

- Function argument type mismatch;
- Function argument value range mismatch;
- Division by zero;
- NULL Pointer reference.

Examples of anticipated error handling are:

- Communication problem handling;
- General IO data processing;
- User input errors.

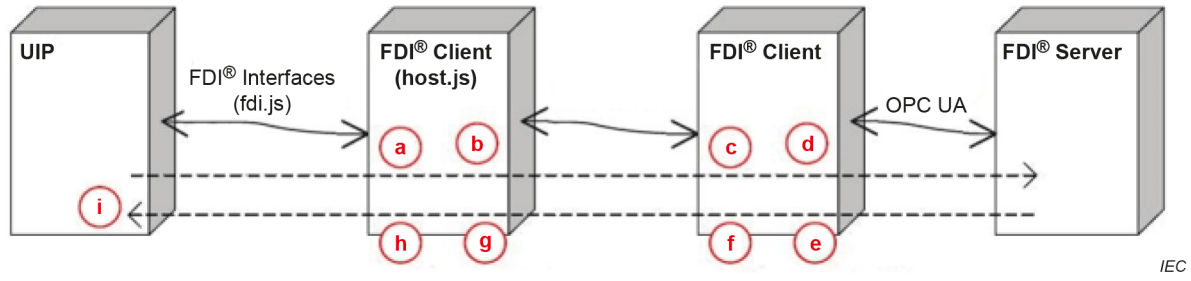


Figure 11 – Component interaction

According to the FDI® Architecture exceptions can occur in different steps of the service processing, see Figure 11:

a) Passing the request from the UIP to the FDI® Client specific service implementation (host.js):

b) Request forwarding from FDI® Client specific service implementation (host.js) to FDI® Client base component;

c) FDI® client base component receives and processes service request

d) Request forwarding to FDI® server;

e) FDI® client base component receives and processes the response from the FDI® Server;

f) Forwarding the response to the FDI® Client specific service implementation (host.js);

g) Response receiving from FDI® client base component and processing;

h) Forwarding the response to UIP;

i) 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 asynchronous pattern the following rules apply.

– Any failure occurring within step a) shall be reported by an error thrown by the call of OperationName and shall be evaluated with .catch()-Handler on the returned Promise.

– Any failure occurring within steps b) to g) shall be reported by an error thrown by the call of OperationName and shall be evaluated with the second .then()-Handler on the returned Promise.

4.6.6 Type safe interfaces

The Information Model hosts device variables of different types. The values of such variables are transferred using the class `Fdi.Model.DataValue`, defined in FDI® Interface definition fdi.ts.

The Device Access Services support writing or reading multiple variables within one service. The data type chosen for data transport is `Fdi.Model.DataValue`. ´Type safe transport is implemented using the `Fdi.Model.DataValue` property `Fdi.Model.Datatype`, which describes the value data type by means of the enumeration `Fdi.Model.Datatype`. Because the `Fdi.Model.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 processing the data.

4.6.7 Globalization and localization

Different Javascript libraries are available to support UIP localization.

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.UIPServices.activate()`.The data type for `currentRegion` is `Fdi.Model.RegionInfo`. The data type for `currentCulture` is `Fdi.Model.CultureInfo`.

4.7 Security

4.7.1 General

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

System wide security begins with the design process, which is out of the standardization scope. From the system perspective security is about controlling access to resources, such as application components, data, and hardware. The HTML5 runtime already constraints access to resources and provides means to control the execution of extensions like plugins.

A complimentary approach is based on certification and authentication. Since any FDI® Package needs compliance testing and certification the presumption is that such certified FDI® Packages don't pose any threats to a system. This means a UIP could be executed with the permissions specified in IEC 62769-2.

While an over-constrained system could lead to functional problems, un-constrained permissions can be seen as a security threat. Thus, subclause 4.7 represents a compromise between both ways.

4.7.2 Access permissions

4.7.2.1 General

Within 4.7.2, 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.

4.7.2.2 Technology specific UIP permissions and restrictions

The general UIP permissions and restrictions are specified in IEC 62769-2. The permissions and restrictions specified in 4.7.2 are specific to HTML5-based UIPs.

a) Launching of Java applets is not allowed.
b) Embedding of flash or shockwave content is not allowed.

The FDI® Client shall restrict the UIP permissions according to this list.

4.7.2.3 Implementation rules

A UIP Variant for the RuntimeId HTML<Version> is executed in the HTML5 runtime. The HTML5 runtime shall implement a sandbox restricting the permissions of the UIP. To control the permissions as specified in IEC 62769-2, the concepts of Content Security Policy Level 2 as specified in W3C CSP2 shall be used. Using Content Security Policies, the HTML5 runtime shall enforce security by the following measures:

- All resources shall be loaded only from the same UIP Package, i.e. the same source as the main entry HTML file of the UIP identfied by <StartElementName>-Attribute of the HTML5 UIP Package. To enforce this, the FDI® Client shall set the following policy: `Content-Security-Policy: default-src 'self'; connect-src 'self' ws://localhost:*>`.
- Any references to external resources shall be ignored by the HTML5 runtime.

- Scripts are allowed in dedicated files only. Script snippets within HTML code are not allowed. All JavaScript code shall be deployed using a separate file and referenced in the HTML file using the <script> tag, as for example <script src="./externalfile.js"></script>.
- By default, cascading style sheets are allowed in dedicated files only. To ensure better interoperability with third party libraries, style definitions within HTML code shall be allowed. This is enforced by setting the following policy in the Content-Security-Policy: `style-src 'self' 'unsafe-inline'`.
- Use of eval() is forbidden and prevented by the HTML5 runtime.
- Event handling shall be done using event listener functions instead of event handler attributes.
- The execution of embedded content as for example Java applets or plugins of the HTML5 runtime is restricted by default when using Content Security Policies. The *plugin-types* directive thus shall not be used. To open a file in the registered default application for a specific MIME type, a UIP can use the hosting service `Fdi.HostingServices.initOpenDefaultApplication`.
- The HTML5 engine shall not provide a standard context menu with back and forward buttons. A UIP may implement its own context menu.

In summary, the resulting security policy shall be as follows: `Content-Security-Policy: default-src 'self';connect-src 'self' ws://localhost:*; style-src 'self' 'unsafe-inline'`.

The UIP developer shall not specify any Content Security Policies in the UIP.

5 Interface definition

The following Tables 1 to 9 specify the mapping between the abstract services specified in IEC 62769-2 and the corresponding Javascript implementation which is found in the library definition file "fdi.ts".

NOTE The File "fdi.ts" can be obtained from FieldcommGroup. (See also http://www.fieldcommgroup.org).

Table 1 specifies the mapping of the Base Property Services.

Table 1 – Base Property Services

Abstract Service	Javascript Implementation
GetDeviceAccessInterfaceVersion	Fdi.BasePropertyServices.getDeviceAccessInterfaceVersion
GetOnlineAccessAvailability	Fdi.BasePropertyServices.getOnlineAccessAvailability

Table 2 specifies the mapping of the Device Model Services.

Table 2 – Device Model Services

Abstract Service	Javascript Implementation
Browse CancelBrowse	Fdi.DeviceModelServices.browse
Read CancelRead	Fdi.DeviceModelServices.read
Write CancelWrite	Fdi.DeviceModelServices.write
CreateSubscription	Fdi.DeviceModelServices.createSubscription
Subscribe	Fdi.DeviceModelServices.subscribe
Unsubscribe	Fdi.DeviceModelServices.unsubscribe
DeleteSubscription	Fdi.DeviceModelServices.deleteSubscription
DataChangeCallback	Fdi.DataChangeCallback.dataChangeCallback[a)]

a) The DataChangeCallback as defined in IEC 62769-2 has to be implemented by the UIP but the Device Model Services interface has to be implemented by the FDI Client. Therefore the DataChangeCallback is defined using a separate interface, Fdi.DataChangeCallback, and not using Fdi.DeviceModelServices interface directly.

Table 3 specifies the mapping of the Access Control Services.

Table 3 – Access Control Services

Abstract Service	Javascript Implementation
InitLock	Fdi.LockingServices.initLock
ExitLock	Fdi.LockingServices.exitLock

Table 4 specifies the mapping of the Direct Access Services.

Table 4 – Direct Access Services

Abstract Service	Javascript Implementation
InitDirectAccess	Fdi.DirectAccessServices.initDirectAccess
ExitDirectAccess	Fdi.DirectAccessServices.exitDirectAccess
Transfer	Fdi.DirectAccessServices.transfer

Table 5 specifies the mapping of the Hosting Services.

Table 5 – Hosting Services

Abstract Service	Javascript Implementation
GetClientTechnologyVersion	Fdi.HostingServices.getClientTechnologyVersion
OpenUserInterface	Fdi.HostingServices.openUserInterface
CloseUserInterface	Fdi.HostingServices.openUserInterface
LogAuditTrailMessage	Fdi.HostingServices.logAuditTrailMessage
SaveUserSettings	Fdi.HostingServices.saveUserSettings
LoadUserSettings	Fdi.HostingServices.loadUserSettings
Trace	Fdi.HostingServices.trace
ShowMessageBox	Fdi.HostingServices.showMessageBox
ShowProgressBar	Fdi.HostingServices.showProgressBar
CancelCallback	Fdi.CancelCallback.cancelCallback[d)]
UpdateShowProgressBar	Fdi.HostingServices.updateShowProgressBar
EndShowProgressBar	Fdi.HostingServices.endShowProgressBar
StandardUIActionItemsChangeCallback	Fdi.HostingServices.standardUIActionItemsChangeCallback
SpecificUIActionItemsChangeCallback	Fdi.HostingServices.specificUIActionItemsChangeCallback
InitExportFile	Fdi.HostingServices.initExportFile
WriteExportFile	Fdi.HostingServices.writeExportFile
FinishExportFile	Fdi.HostingServices.finishExportFile
InitImportFile	Fdi.HostingServices.initImportFile
ReadImportFile	Fdi.HostingServices.readImportFile
FinishImportFile	Fdi.HostingServices.finishImportFile
InitOpenDefaultApplication	Fdi.HostingServices.initOpenDefaultApplication
WriteOpenDefaultApplication	Fdi.HostingServices.writeOpenDefaultApplication
FinishOpenDefaultApplication	Fdi.HostingServices.finishOpenDefaultApplication
GetEnvironmentProperties	Fdi.HostingServices.getEnvironmentProperties
	Fdi.HostingServices.registerUIPServices[c)]

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

c) The service Fdi.HostingServices.registerUIPService is a technology dependent service, which is not defined in IEC 62769-2. It shall be the first service to be called by the UIP on its activation process to give the FDI Client access to UIP Service interface functions.

d) The service CancelCallback as defined in IEC 62769-2 has to be implemented by the UIP but the Hosting Services interface has to be implemented by the FDI Client. Therefore the CancelCallback is defined using a separate interface, Fdi.CancelCallback, and not using Fdi.HostingServices interface directly.

Table 6 specifies the mapping of the UIP Services.

Table 6 – UIP Services

Abstract Service	Javascript Implementation
Activate	Fdi.UIPServices.activate
Deactivate	Fdi.UIPServices.deactivate
SetSystemLabel	Fdi.UIPServices.setSystemLabel
SetTraceLevel	Fdi.UIPServices.setTraceLevel
InvokeStandardUIAction	Fdi.UIPServices.invokeStandardUIAction
InvokeSpecificUIAction	Fdi.UIPServices.invokeSpecificUIAction
GetStandardUIActionItems	Fdi.UIPServices.getStandardUIActionItems
GetSpecificUIActionItems	Fdi.UIPServices.getSpecificUIActionItems

Table 7 specifies the mapping of the base data types.

Table 7 – Base Data Types

Base data type	Javascript Implementation
Boolean	Fdi.Model.Datatype.Boolean
String	Fdi.Model.Datatype.String
ByteString	Fdi.Model.Datatype.Binary
UtcTime	Fdi.Model.Datatype.DateTime
Int8	Fdi.Model.Datatype.SByte
Int16	Fdi.Model.Datatype.Short
Int32	Fdi.Model.Datatype.Int
Int64	Fdi.Model.Datatype.Long
Byte	Fdi.Model.Datatype.Byte
UInt16	Fdi.Model.Datatype.UShort
UInt32	Fdi.Model.Datatype.UInt
UInt64	Fdi.Model.Datatype.ULong
Float	Fdi.Model.Datatype.Float
Double	Fdi.Model.Datatype.Double
Duration	Fdi.Model.Datatype.TimeSpan

Table 8 specifies the mapping of the special data types.

Table 8 – Special Types

Special Data type	Javascript Implementation
Attribute Ids	Fdi.Model.AttributeType
Variant	Fdi.Model.Variant
NodeSpecifier	Fdi.Model.NodeSpecifier
Data Value	Fdi.Model.DataValue
Localized Text	Fdi.Model.Datatype.LocalizedText
Range	Fdi.Model.Datatype.Range
EU Information	Fdi.Model.Datatype.EUInformation
Enum Value	Fdi.Model.Datatype.EnumValueType
InnerErrorInfo	Fdi.Model.InnerErrorInfo
NumericRange	Fdi.Model.ArrayIndexRange

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

Table 9 specifies the mapping of the parameter types.

Table 9 – Parameter Types

Special Data type	Javascript Implementation
TraceLevel	Fdi.Model.TraceLevel
StandardUIAction	Fdi.Model.StandardUIAction
StandardUIActionItem	Fdi.Model.StandardUIActionItem
SpecificUIActionItem	Fdi.Model.SpecificUIActionItem

Detailed interface definition and interface documentation are available in:

- fdi.ts (Typescript file), fdi.js (transpiled fdi.ts) and accompanying documentation

Bibliography

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

ECMA-262, *ECMAScript® Language Specification*

W3C CSP2, *W3C Recommendation Content Security Policy Level 2*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTÉGRATION DES APPAREILS DE TERRAIN (FDI®) –

Partie 6-200: Mapping de technologies – HTML5

AVANT-PROPOS

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

L'IEC 62769-6-200 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. Il s'agit d'une Norme internationale.

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

Projet	Rapport de vote
65E/870/CDV	65E/926/RVC

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

La langue employée pour l'élaboration de cette Norme internationale est l'anglais.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2, il a été développé selon les Directives ISO/IEC, Partie 1 et les Directives ISO/IEC, Supplément IEC, disponibles sous www.iec.ch/members_experts/refdocs. Les principaux types de documents développés par l'IEC sont décrits plus en détail sous www.iec.ch/publications.

Une liste de toutes les parties de la série IEC 62769, publiées sous le titre général *Intégration des appareils de terrain (FDI®)*, se trouve 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 webstore.iec.ch dans les données relatives au document recherché. A 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 ce document indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer ce document en utilisant une imprimante couleur.

INTÉGRATION DES APPAREILS DE TERRAIN (FDI®) –

Partie 6-200: Mapping de technologies – HTML5

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®[1], *Field Device Integration*). Le mapping de technologies porte essentiellement sur la mise en œuvre des composants Client FDI® et Plugiciel d'interface utilisateur (UIP, *User Interface Plug-in*) à l'aide de l'environnement d'exécution HTML5.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences 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).

FCG TS10099, *Field Device Integration (FDI®) – Technology Management* (disponible en anglais seulement)

IEC 62769-1:2021, *Intégration des appareils de terrain (FD®I) – Partie 1: Vue d'ensemble*

IEC 62769-2:2021, *Intégration des appareils de terrain (FDI®) – Partie 2: Client*

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

IEC 62769-6-100, *Intégration des appareils de terrain (FDI®) – Partie 6-100: Mapping de technologies – .NET*

W3C HTML5.0, *W3C Recommendation HTML5 – A vocabulary and associated APIs for HTML and XHTML* (disponible en anglais seulement)

3 Termes, définitions, abréviations, acronymes et conventions

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC 62769-1 et l'IEC 62769-6-100 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

---

[1] FDI est une marque déposée de l'organisation à but non lucratif Fieldbus Foundation, Inc. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que l'IEC approuve le détenteur de la marque ou l'emploi de ses produits. La conformité n'exige pas l'utilisation de la marque. L'utilisation de la marque exige l'autorisation du détenteur de la marque.

3.1.1
Environnement d'exécution HTML5
composant fonctionnel du Client FDI®, qui exécute l'UIP HTML5 ;en principe, il fournit la fonctionnalité d'un navigateur web (rendu HTML et moteur d'exécution JavaScript)

3.1.2
Bibliothèque de types d'hôtes FDI®
mise en œuvre spécifique à l'hôte de la Bibliothèque de types FDI® qui comporte host.js et fdi.js. La Bibliothèque de types d'hôtes FDI® est mise à disposition par les fournisseurs d'hôtes et est déployée avec l'UIP au démarrage de celui-ci

3.2 Abréviations et acronymes

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

UML (Unified Modeling Language) Langage de modélisation unifié

3.3 Conventions

Les figures du présent document utilisent des symboles graphiques conformément à l'ISO/IEC 19505-1 (UML 2.0).

4 Concepts techniques

4.1 Généralités

4.1.1 Vue d'ensemble

En 4.1.2, le présent document décrit la base technologique de la mise en œuvre de l'UIP qui repose sur HTML5, ainsi que l'environnement logiciel, y compris les règles connexes de mise en œuvre. L'Article 4 suit une approche orientée cycle de vie (cas d'utilisation).

Le 4.3.4 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 l'achèvement de l'UIP sont décrits en 4.5. Le 4.6 définit les règles d'interaction entre le Client FDI® et l'UIP. Les définitions relatives à la sécurité sont données en 4.7. Les définitions d'interface de service pour le Client FDI® et l'UIP sont spécifiées à l'Article 5.

4.1.2 Bibliothèque de types FDI®

Les Services d'accès à l'Appareil, les Services d'hébergement et les Services d'UIP peuvent être modélisés comme des interfaces TypeScript qui transmettent des arguments de types de données TypeScript. Ces interfaces et ces types de données sont utilisés pour l'échange de données et l'interaction entre l'UIP et le Client FDI®. Afin de gérer les erreurs d'exécution durant les appels de méthode d'interface, des objets d'erreur TypeScript sont définis.

Les interfaces, types de données et classes d'exception FDI® TypeScript sont définis dans une Bibliothèque de types FDI® unique. Le nom de fichier de la définition d'interface doit être "fdi.ts".

La Figure 1 représente la structure de la Bibliothèque de types FDI®.

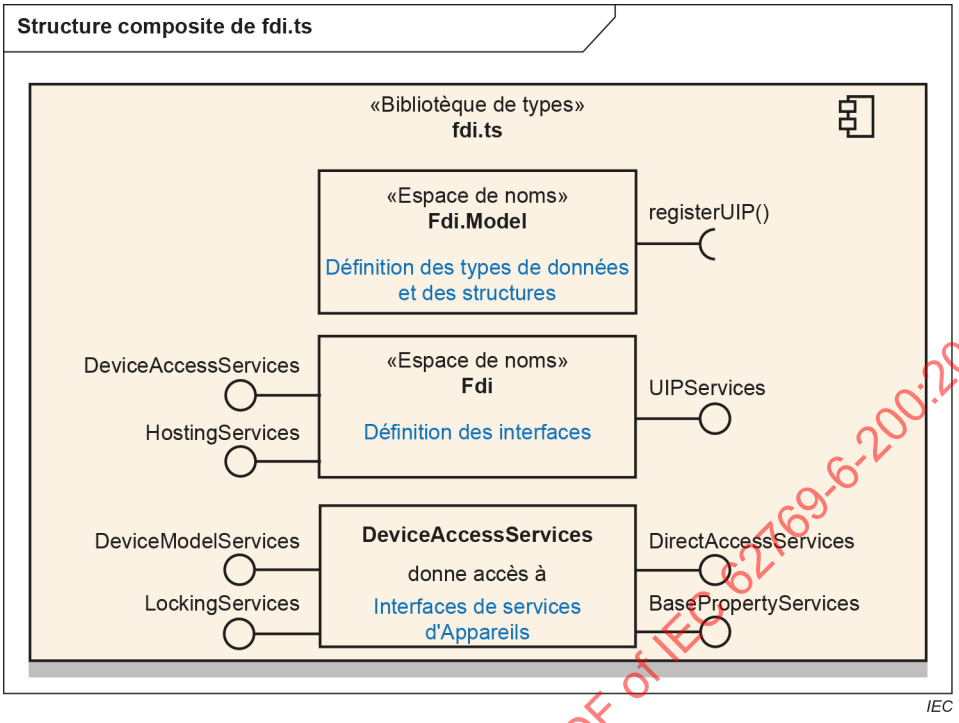


Figure 1 – Structure de la Bibliothèque de types FDI®

NOTE 1 Le diagramme de la structure composite ne représente que les interfaces de base qui mettent en œuvre les interfaces définies dans l'IEC 62769-2.

Les interfaces doivent être mises en œuvre par le Client FDI® et l'UIP, respectivement, à l'aide de TypeScript. Les fichiers de mise en œuvre TypeScript sont transpilés en JavaScript à l'aide de l'ensemble de fonctionnalités spécifié dans l'édition de l'ECMA 262, qui correspond à l'identificateur d'exécution (RuntimeId) de la variante d'UIP. L'édition de l'ECMA 262 est explicitement spécifiée dans la FCG TS10099 par le RuntimId de la variante d'UIP.

La transpilation de fdi.ts donne fdi.js, qui contient les prototypes des fonctions. La Bibliothèque de types FDI® (fdi.js) combinée à la mise en œuvre des interfaces spécifique à l'hôte (host.js) est appelée Bibliothèque de types d'hôtes FDI®. La structure globale de la Bibliothèque de types FDI® ainsi que la mise en œuvre des interfaces spécifique à l'hôte et à l'UIP sont représentées à la Figure 2.

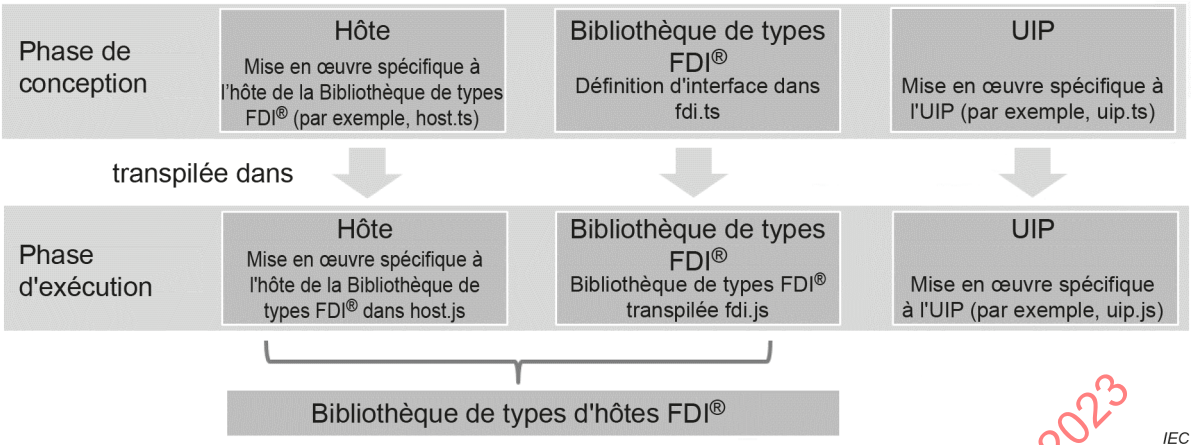


Figure 2 – Relation entre la Bibliothèque de types FDI® et la mise en œuvre des interfaces spécifique à l'hôte et à l'UIP

NOTE 2 La Bibliothèque de types FDI® est définie dans cette spécification et peut être obtenue auprès de FieldcommGroup (voir www.fieldcommgroup.org). La mise en œuvre spécifique à l'hôte est proposée par les fournisseurs d'hôtes et la mise en œuvre spécifique à l'UIP fait partie de la variante d'UIP particulière.

Les Bibliothèques de types FDI® fdi.ts et fdi.js, respectivement, doivent être versionnées conformément à l'IEC 62769-1:2021, 8.1. La Bibliothèque de types FDI® fait partie de la Technologie FDI® de base, conformément à l'IEC 62769-1:2021, 8.3.2.1, et a donc une incidence directe sur la Version de Technologie FDI®. Toutes les modifications compatibles des fichiers fdi.ts et fdi.js, respectivement, se traduisent par une augmentation de la partie mineure de la Version de Technologie FDI®. Les modifications incompatibles se traduisent par une augmentation de la partie majeure de la Version de Technologie FDI® (voir l'IEC 62769-1:2021, 8.3.2.2).

La Bibliothèque de types FDI® doit être installée lors de chaque installation de Client FDI®. Les Plugiciels d'interface utilisateur (UIP) et l'Application du Client FDI® doivent utiliser cette instance du fichier fdi.js. 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 mise en œuvre transpilée spécifique à l'hôte de la Bibliothèque de types FDI® (host.js) et la Bibliothèque de types FDI® proprement dite (fdi.js) doivent être fournies par le Client FDI® lors du processus de chargement de l'UIP. Les deux fichiers doivent se trouver dans le sous-dossier "./scripts" du dossier d'installation de l'UIP. Les fichiers respectifs fdi.js et host.js de la Bibliothèque de types d'hôtes FDI® doivent être explicitement référencés par l'UIP. Par exemple, ce référencement peut être réalisé à l'aide des énoncés <script src="/scripts/fdi.js" type="module"/> et <script src="./scripts/host.js" type="module"/> dans l'énoncé <header> du fichier HTML nommé d'après l'Attribut <StartElementName> du Paquetage de l'UIP HTML5.

4.2 Représentation de l'UIP

4.2.1 Autonomie

La Variante d'UIP peut contenir un ou plusieurs fichiers HTML et leurs fichiers supplémentaires connexes (fichiers de ressources, par exemple). Le fichier HTML d'entrée principal est directement placé dans le répertoire d'installation de l'UIP. Son nom lui est donné par l'attribut `<StartElementName>` du Paquetage UIP HTML5.

Outre la Bibliothèque de types d'hôtes FDI® (fdi.js et host.js), la variante d'UIP doit être autonome. Tous les fichiers supplémentaires exigés par l'UIP sont stockés dans un ou plusieurs sous-dossiers du répertoire d'installation. La Bibliothèque de types d'hôtes FDI® est fournie par le Client FDI®.

La Figure 3 représente un exemple d'UIP avec des sous-dossiers pour les scripts, des feuilles de style en cascade (CSS) et des images.

```
async function browseParmeterList() {

    var node = new UIPTypes.NodeSpecifier("parameterSet", false);                    // create NodeSpecifier to be browsed
    var cancelToken = new UIPTypes.CancelToken(guid());                              // create cancel token to enable cancelling of call
    tokenDirectory[cancelToken.id] = cancelToken;

    let fdiDMS = deviceModelServices;                                                // interface to host specific implementation of device model services

    await fdiDMS.browse(node,cancelToken).then(                                      // async call to browse function
        response => { BrowseResponse(response, "Browse"); },                         // response evaluation, if call returns successful
        error => {                                                                   // response evaluation, if call returns wit failure
            if(error == Fdi.Model.StatusCode.Bad_RequestCancelled){                  // call was cancelled
                log("async call to Browse with ID:" + cancelToken.id + " cancelled");
                delete tokenDirectory[cancelToken.id];
            }
            else{                                                                    // other error inside browse function call
                log("async call to Browse with ID:" + cancelToken.id + " failed");
                delete tokenDirectory[cancelToken.id];
            }
        }
    ).catch(                                                                         // call to browse service failed
        exception => { log("async call to Browse with ID:" + cancelToken.id + " catched");
            delete tokenDirectory[cancelToken.id];
        }
    );
};
```

IEC

Figure 3 – Exemples de fichiers de données supplémentaires de l'UIP, et notamment de fichiers JavaScript, de fichiers CSS et d'images

4.2.2 Bibliothèques externes et fichiers supplémentaires

L'UIP peut s'appuyer sur des bibliothèques externes (bibliothèques tierces) et d'autres composants, comme des bibliothèques de contrôle d'utilisateur spécifiques, des icônes ou des images. Les Clients FDI® chargent et exécutent l'UIP dans l'environnement d'exécution HTML5 de manière dynamique. Pour prendre en charge cette utilisation, ainsi que l'exigence de prévenir les éventuels problèmes liés à des bibliothèques en conflit, les règles suivantes sont spécifiées pour les bibliothèques externes et les fichiers supplémentaires.

Les bibliothèques externes et les fichiers supplémentaires:

- doivent être contenus dans la Variante d'UIP, à l'exception de la Bibliothèque de types d'hôtes FDI® (fdi.js et host.js) qui est fournie par le Client FDI®;
- ne doivent pas exiger d'autres procédures d'installation que leur copie dans le répertoire d'installation de l'UIP ou dans ses sous-dossiers;
- doivent respecter les restrictions d'accès décrites en 4.7.2;
- doivent être compatibles avec les plateformes et les environnements d'exécution pour lesquels la Variante d'UIP est conçue.

Le Client FDI® charge l'UIP HTML5 en accédant au fichier d'entrée principal donné par l'Attribut `<StartElementName>` du Paquetage de l'UIP HTML5. Les bibliothèques externes et les fichiers supplémentaires référencés doivent être explicitement chargés par l'UIP lui-même (ce qui déclaré sous la forme <script src="./scripts/myExternalLibrary.js"/>, par exemple). Toutes les références à des bibliothèques externes et à des fichiers supplémentaires dans l'UIP doivent être spécifiées sous la forme d'URL relatives.

4.3 Règles de compatibilité de l'UIP

4.3.1 Vue d'ensemble

Les règles de compatibilité pour les différentes versions de la variante d'UIP sont spécifiées dans l'IEC 62769-4.

Un Client FDI® doit fournir les versions de la Bibliothèque de types d'hôtes FDI® (fdi.js et host.js) ainsi que l'environnement d'exécution HTML5 qui prend en charge l'ensemble de fonctionnalités, lesquels correspondent au RuntimeId de la variante d'UIP.

4.3.2 Stratégie de compatibilité de la Bibliothèque de types FDI®

L'UIP doit vérifier la Version de Technologie FDI® du Client FDI® à l'aide du HostingService GetClientTechnologyVersion. Les règles suivantes assurent la compatibilité de l'UIP avec la Bibliothèque de types d'hôtes FDI®:

- Si la variante d'UIP a été créée pour une Version de Technologie FDI® totalement compatible (versions majeures et mineures identiques), l'UIP peut être démarré directement dans le Client FDI®.
- Si la version mineure de la Technologie FDI® du Client FDI® est antérieure à la version mineure de la Technologie FDI pour laquelle l'UIP a été créé, l'UIP peut uniquement utiliser les méthodes d'interface définies dans la Version de Technologie FDI® prise en charge par le Client FDI®.
- Si la variante d'UIP à démarrer a été créée pour une Version de Technologie FDI® incompatible (versions majeures distinctes), le Client FDI® doit démarrer l'UIP à l'aide de la Bibliothèque de types d'hôtes FDI® et de l'environnement d'exécution correspondants.

4.3.3 Stratégie de compatibilité des environnements d'exécution

Dans différentes versions d'environnement d'exécution, les UIP peuvent utiliser un ensemble de fonctionnalités différent, potentiellement incompatibles. L'ensemble de fonctionnalités, qui doit être pris en charge par toute version d'environnement d'exécution identifiée par un RuntimeId particulier, est spécifié dans la FCG TS10099.

Les Clients FDI® qui prennent en charge un RuntimeId particulier doivent prendre en charge les fonctionnalités répertoriées dans la FCG TS10099, pour chaque RuntimeId au minimum.

Les variantes d'UIP conçues pour un RuntimeId particulier peuvent s'appuyer sur les fonctionnalités spécifiées dans la FCG TS10099 pour ce RuntimeId. Si un UIP utilise des fonctionnalités qui ne sont pas répertoriées dans l'ensemble de fonctionnalités normalisé du RuntimeId correspondant, l'UIP doit vérifier la disponibilité de la fonctionnalité dans le Client FDI® concerné et mettre en œuvre une stratégie de repli, par exemple un message d'information envoyé à l'utilisateur.

Si un Client FDI® détecte qu'une Variante d'UIP est conçue pour un RuntimeId qui exige un autre ensemble de fonctionnalités de l'environnement d'exécution, ce Client FDI® peut utiliser l'environnement d'exécution approprié, par exemple l'environnement d'exécution HTML5+, pour exécuter l'UIP (voir Figure 4).

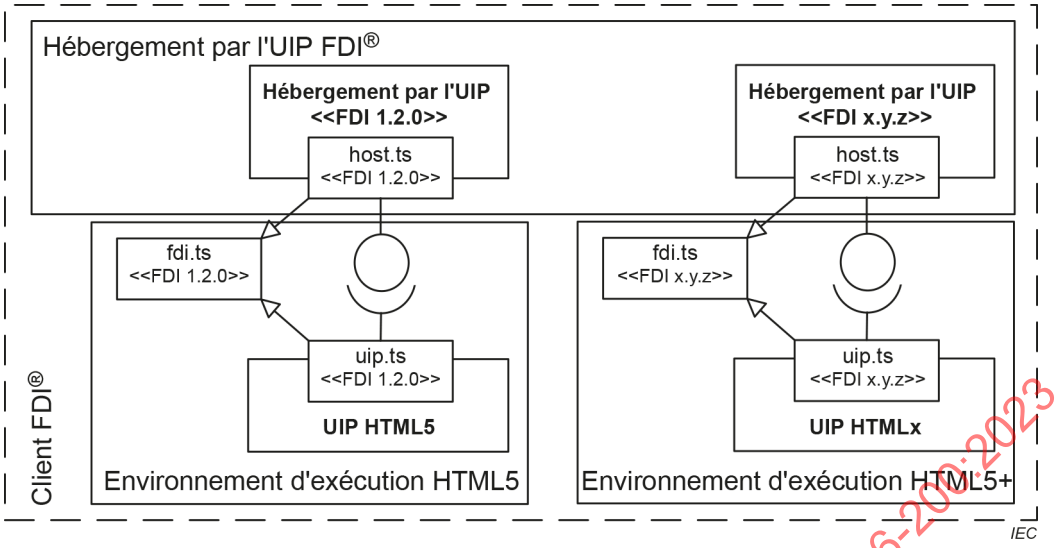


Figure 4 – Stratégie de compatibilité pour héberger des UIP conçus pour des versions d'environnement d'exécution distinctes

4.3.4 Règles de compatibilité des exécutables de l'UIP

Un Plugiciel d'interface utilisateur HTML5 n'est pas compilé en code machine. L'élément "CpuInformation" du fichier catalog.xml, qui contient les informations relatives à la plateforme de compilation cible, ne doit pas être utilisé pour les Plugiciels d'interface utilisateur HTML5.

4.4 Déploiement de l'UIP

Les règles générales d'installation de l'UIP sont décrites dans l'IEC 62769-2. Si l'UIP est stocké dans le système de fichiers local, le Client FDI® et le Serveur FDI® doivent assurer l'intégrité de l'UIP.

La mise en œuvre du Client FDI® permet de s'assurer que le déploiement de l'UIP fonctionne indépendamment des informations d'identification de l'utilisateur actuel (voir la note ci-dessous).

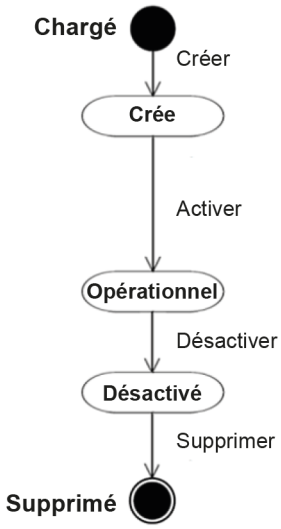
NOTE Certains dossiers gérés par le système d'exploitation exigent des droits d'accès spécifiques, par exemple, des modifications dans le dossier "Programmes" exigent des droits "Administrateur". Le système d'exploitation Windows fournit plusieurs moyens qui permettent à une application en cours d'exécution 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 méthode alternative consiste à copier l'UIP dans des dossiers inscriptibles pour les utilisateurs "ordinaires".

4.5 Cycle de vie de l'UIP

4.5.1 Généralités

Le diagramme d'états de l'UIP, décrit dans l'IEC 62769-2, est composé des états Loaded, Created, Operational, Deactivated et Disposed. Les mécanismes qui ont une incidence sur les changements d'états sont décrits en 4.5.

Après le déploiement de l'UIP (voir 4.3.4), le Client FDI® charge l'environnement d'exécution HTML5 de manière dynamique dans la mémoire et exécute la logique associée de l'UIP en appelant les fonctions d'interface FDI® spécifiées correspondantes. La Figure 5 fournit une vue d'ensemble du cycle de vie de l'UIP.



Étape de transition	État de la source	État de la destination	Description
Créer	Chargé	Crée	Le Client FDI® charge son environnement d'exécution HTML5, créé un contexte de navigation pour l'UIP et parcourt ce contexte jusqu'à la page d'entrée HTML principale indiquée par l'Attribut <StartElementName> du Paquetage de l'UIP HTML5. Toutes les ressources référencées sur la page d'entrée HTML principale doivent être chargées, comme cela est spécifié dans W3C HTML5.
Activer	Crée	Opérationnel	L'environnement d'exécution HTML5 envoie l'événement Javascript "onload" à l'UIP. L'UIP doit lier sa méthode d'activation à cet événement. Dans cette méthode, il met en œuvre sa séquence de démarrage et d'initialisation qui inclut l'enregistrement de la mise en œuvre de l'interface UIPServices sur le Client FDI® en appelant la méthode Fdi.Model.registerUIP(). Enfin, le Client FDI® appelle la méthode UIPServices.activate() et l'UIP passe à l'état "Opérationnel"
Désactiver	Opérationnel	Désactivé	Le Client FDI® doit appeler la méthode UIPServices.deactivate() en utilisant l'interface UIPServices enregistrée. L'UIP doit achever ses opérations et stocker les données. L'UIP doit se placer dans un état où il peut être libéré de manière sûre.
Supprimer	Désactivé	Supprimé	Le Client FDI® doit supprimer l'ensemble des références à l'instance de l'UIP.

IEC

Figure 5 – Cycle de vie de l'UIP

Les 4.5.2 et 4.5.3 spécifient les règles qui définissent comment le Client FDI® doit activer et désactiver l'UIP.

4.5.2 Etapes d'activation de l'UIP

4.5.2.1 Load (charger)

Le Client FDI® doit charger son environnement d'exécution HTML5. Dans l'environnement d'exécution HTML5, la création de l'UIP doit être réalisée en créant un contexte de navigation pour l'UIP et en parcourant ce contexte de navigation jusqu'à la page d'entrée HTML principale indiquée par l'Attribut `<StartElementName>` du Paquetage de l'UIP HTML5. L'Attribut `<StartElementName>` est contenu dans le fichier uipcatalog.xml du Paquetage de l'UIP.

4.5.2.2 Create (créer)

L'UIP est créé dans l'environnement d'exécution HTML5. Toutes les ressources référencées sur la page d'entrée HTML principale doivent être chargées, comme cela est spécifié dans W3C HTML5.0.

Lorsque l'UIP a été créé, l'environnement d'exécution HTML5 doit envoyer l'événement JavaScript "onload" à l'UIP.

4.5.2.3 Activate (Activer)

L'UIP doit lier sa méthode de démarrage à l'événement JavaScript "onload". Dans le cadre de cette méthode, l'UIP met en œuvre sa séquence de démarrage et d'initialisation.

Pour cette méthode de démarrage, l'UIP doit appeler la méthode `Fdi.Model.registerUIP()` pour permettre au Client FDI® d'appeler les méthodes sur l'UIP comme cela est spécifié en 6.1, Services UIP de l'IEC 62769-2:2021.

Après l'enregistrement de la mise en œuvre de l'interface des Services d'UIP de l'UIP particulière, le Client FDI® doit appeler la méthode `Fdi.UIPService.setSystemLabel()` pour spécifier un identificateur en langage humain de l'instance de l'UIP dans le contexte du Client FDI®. Afin d'achever l'activation de l'UIP, le Client FDI® appelle `Fdi.UIPService.activate()`. L'appel de `Fdi.UIPService.activate()` inclut des informations sur la localisation (RegionInfo, CultureInfo) et le contexte d'appel, notamment sur la mise en œuvre des interfaces DeviceAccessServices and HostingServices propre au Client FDI®. Lorsque `Fdi.UIPService.activate()` s'est achevé avec succès, le processus d'activation de l'UIP est terminé et l'UIP est opérationnel.

L'ensemble du processus d'activation est représenté à la Figure 6.

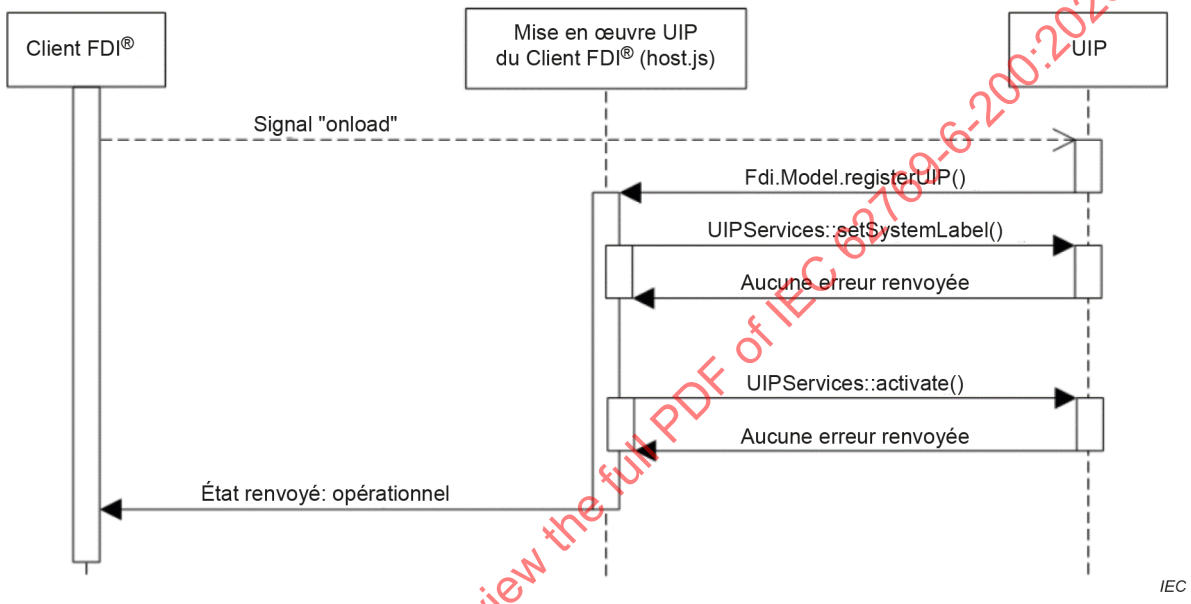


Figure 6 – Processus d'activation de l'UIP

4.5.3 Désactivation de l'UIP

Pour désactiver l'UIP, le Client FDI® doit appeler la méthode `Fdi.UIPService.deactivate()` à l'aide de la mise en œuvre de l'interface de Services d'UIP enregistrée lors du processus d'activation.

Le Client FDI® doit ensuite supprimer toutes les références à l'instance de l'UIP.

4.5.4 Règles générales relatives à l'activation et à la désactivation de l'UIP

La logique d'activation doit être limitée pour instancier l'objet en ce qui concerne la structure de données interne et la charge de toutes les bibliothèques référencées. La mise en œuvre de la désactivation doit être limitée pour détruire l'objet en ce qui concerne les ressources de mémoire de publication. L'activation et la désactivation ne doivent pas:

- envoyer des erreurs;
- envoyer de rappel au Client FDI®; ni
- appeler d'interaction utilisateur.

4.6 Interaction entre un Client FDI® et un UIP

4.6.1 Traitement des éléments d'UI normalisés

Les UIP doivent déléguer au Client FDI® la présentation et le traitement des éléments d'UI normalisés. Ces éléments d'UI normalisés sont:

- les Actions d'UI avec la sémantique normalisée (Appliquer/Fermer/Aide en ligne); et
- les Actions d'UI spécifiques (Actions spécifiques à un IUP, par exemple Réinitialiser, Reconnecter).

Pour assurer des interactions cohérentes de l'Interface utilisateur dans les UIP 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 peuvent mettre en œuvre des actions d'UI non normalisées dans leur propre partie UI.

L'ensemble des actions d'UI normalisées 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 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 méthodes `Fdi.HostingServices.StandardActionItemSetChanged()` et `Fdi.HostingServices. ApplicationSpecificActionItemSetChanged()`).

Un Client FDI® peut utiliser des éléments d'UI spécifiques, par exemple les contrôles de boutons, pour fournir un accès direct aux actions normalisées, mais également pour les appeler 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 d'UI spécifiques.

4.6.2 Exécution d'un service sans blocage

4.6.2.1 Fonctions internes du Client FDI®

La partie productive (chronophage) d'un service nommé *NomOpération* doit être exécutée de manière asynchrone par le Client FDI®. Pour l'exécution d'un service asynchrone, le modèle async/await fondé sur des objets Promise est utilisé. Les méthodes de mise en œuvre de l'interface FDI® renvoient donc un objet Promise (voir Figure 7).

Si la demande ne peut pas être exécutée, le Client FDI® doit émettre l'objet Promise avec "reject" pour indiquer que l'opération demandée n'a pas démarré. Toute autre réponse doit émettre l'objet "Promise" avec "resolve". La valeur de retour fournie avec "resolve" doit inclure un message d'erreur approprié et l'un des codes de statut définis avec `Fdi.Model.StatusCode` si l'opération qui prend du temps a engendré une erreur ou a été annulée.

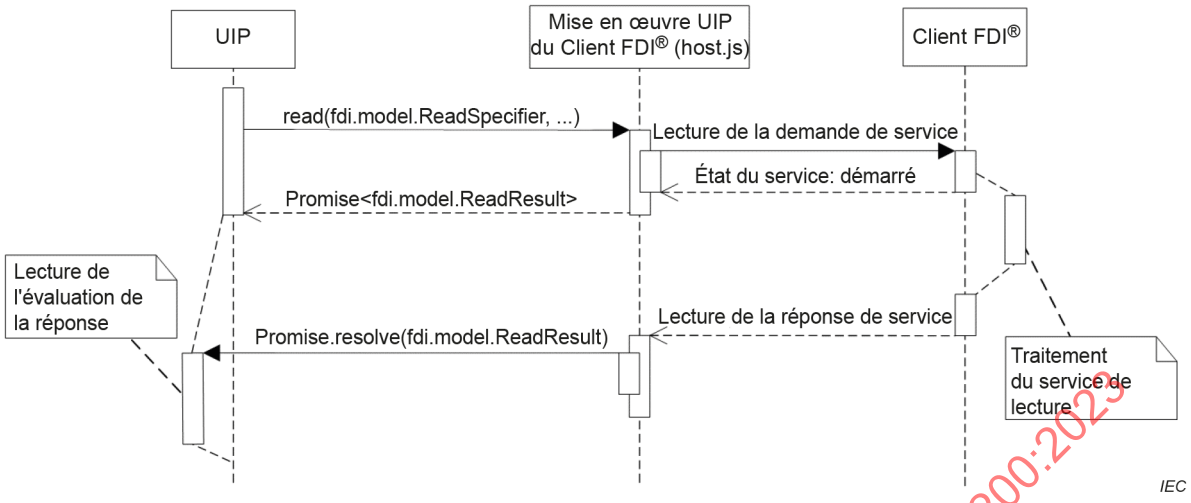


Figure 7 – Exemple d'exécution d'un service asynchrone (Client FDI®)

La mise en œuvre de Cancel*NomOpération* n'est pas effectuée en tant qu'appel de service distinct. A la place, les appels de service qui peuvent être annulés ont un argument `cancelToken` de type `Fdi.Model.CancelToken`. Si l'UIP annule l'exécution du service, il appelle la méthode `cancelToken.cancel()` qui doit appeler la résolution de l'objet Promise avec `Fdi.Model.StatusCode.Bad_RequestCancelled` et un message d'erreur approprié (voir Figure 8).

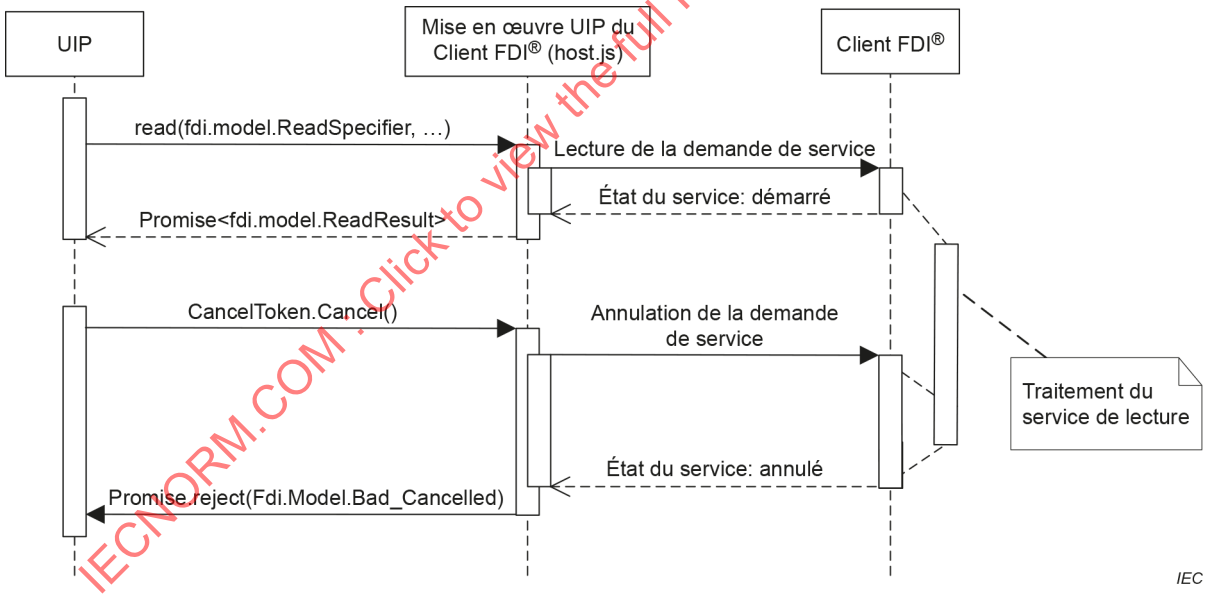


Figure 8 – Exemple d'exécution d'un service asynchrone annulé (Client FDI®)

Le traitement de l'opération dans l'UIP est décrit dans le 4.6.2.2 ci-après.

4.6.2.2 Fonctions internes de l'UIP

La mise en œuvre de chaque fonction d'interface de service qui est appelée à partir de l'UIP renvoie un objet asynchrone de classe "Promise<RésultatOpération>". L'UIP évalue l'objet Promise en mettant en œuvre son gestionnaire ".then". En outre, il convient de mettre en œuvre le gestionnaire ".catch" pour détecter les erreurs inattendues. Voir l'exemple représenté à la Figure 9.

Chaque gestionnaire ".then" doit avoir deux fonctions asynchrones, une pour "resolve" et une pour "reject". Une opération qui engendre un "resolve" a été exécutée avec succès. Une opération qui engendre un "reject" a été annulée ou l'exécution du service dans le Client FDI® a généré une erreur. Le gestionnaire ".catch" traite en outre les exceptions qui peuvent survenir dans le cadre de la mise en œuvre spécifique à l'hôte (conversions de types ayant échoué ou arguments manquants, par exemple).

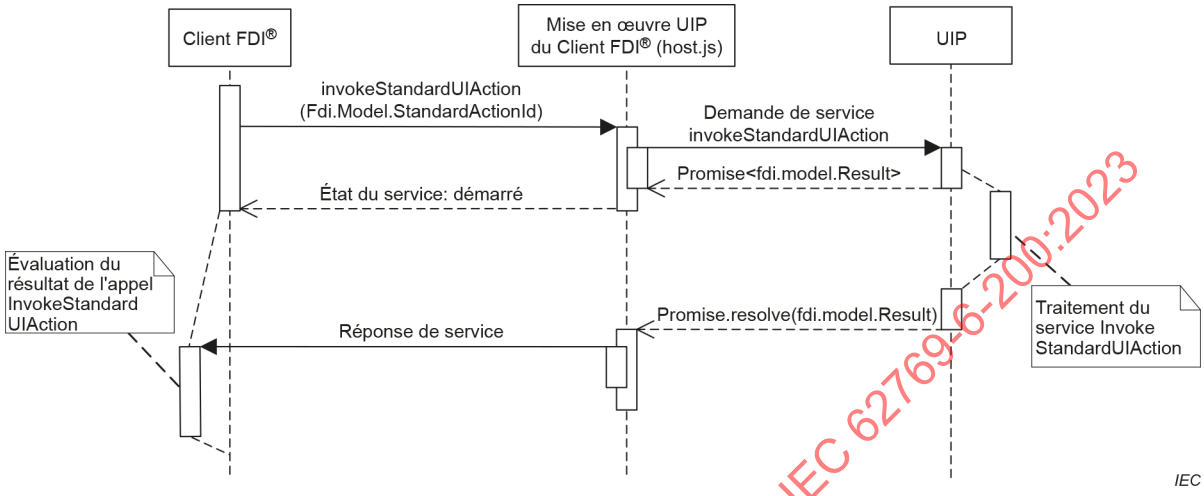


Figure 9 – Exemple d'exécution d'un service asynchrone (UIP)

L'utilisation de Promise implique que la fonction d'interface correspondante est appelée à partir d'une fonction déclarée avec le mot clé "async", comme cela est indiqué dans l'exemple de code à la Figure 10. Le mot clé "async" permet le traitement asynchrone de la fonction. La fonction d'interface elle-même doit être appelée avec l'énoncé "await". Toute mise en œuvre dans le gestionnaire ".then()" ou ".catch()" doit être exécutée dans un fil distinct par le moteur HTML5 et ne doit donc pas bloquer l'interface utilisateur.

```
async function OnBrowseButton() {                                    // event handler using 'async' keyword

  var fdiDMS = new DeviceModelServices();                            // interface pointer to Device Model Services
  var node = new Fdi.Model.NodeSpecifier("/ParameterSet/Tag", true); // definition of node to browse

  var cancelTokenId = guid();
  var cancelToken = new Fdi.Model.CancelToken(cancelTokenId);        // creation of a cancel token

  await fdiDMS.browse(node, cancelToken)                             // browse service call
    .then(
      response => { EvalBrowseResponse(response, "Browse"); },       // first .then handler to cover 'resolve' of service
      error => {                                                     // second .then handler to cover 'reject' of service
        if (error.state == Fdi.Model.StatusCode.Bad_RequestCancelled) // handling of cancelled services
          log("async call to Browse cancelled.");
        else                                                         // common failure handling
          log("async call to Browse returned with failure.");

  }).catch(                                                          // service start fails, service not executed
    exception => {
      log("async call to Browse not executed.");
  });
};
```

IEC

Figure 10 – Extrait de code simplifié pour représenter la mise en œuvre de fonction décrite

4.6.3 Threading (gestion de fils)

4.6.3.1 Règles de mise en œuvre

L'UIP et le Client FDI® ne doivent pas bloquer le fil de l'Interface utilisateur. L'Interface utilisateur doit toujours rester réactive.