

A large, light blue circular graphic with a grey swoosh trailing from its bottom left, positioned behind the title text.

# Navigate Pro – Partner Extension

---

**Info Studi S.r.l.**

Via L. Ariosto 21 – 20091 Bresso (MI)  
Tel. +39 02 6103411 – Fax +39 02 61034190  
[www.infostudi.it](http://www.infostudi.it) - [infostudi@infostudi.it](mailto:infostudi@infostudi.it)

**Sede di Vicenza**

Via Istria 6/8 – 36023 Longare (VI)  
Tel. +39 0444 1805350 – Fax +39 02 61034190

P.IVA 00924450968 - C.F. 08764250158

REA MI 1248744 - Registro Imprese MI149-36561 Capitale sociale Euro 52.000,00 i.v.

## Sommario

|                                                    |   |
|----------------------------------------------------|---|
| Abstract .....                                     | 3 |
| Adding Navigate Pro as a dependency .....          | 4 |
| Subscribing Navigate Pro Button on Documents ..... | 5 |
| Json data structures.....                          | 6 |
| Document part.....                                 | 6 |
| Relationship Part.....                             | 7 |
| Events .....                                       | 9 |

## Abstract

---

In this document you can find a method, to submit your own data to Navigate Pro graphical engine.

The main goal of this document is to show partner:

- How to extend Navigate Pro app;
- How to subscribe to events, skipping the original ones;
- How to send a new set of data to navigate pro engine.

All the examples are referred to a demo a codeunit, "codeunit 50147 PartnerNavigateProMgt", that shows a way to interface the original app.

All the images are related to a Visual Studio Code (VSC) project having runtime equal to 7.

```
"runtime": "7.0"
```

## Adding Navigate Pro as a dependency

---

To extend Navigate Pro, you need to add the original app in your *app.json* file.

```
"dependencies": [{  
  "id": "bf243966-fa6b-4ede-8bc6-25c17b379f46",  
  "publisher": "Info Studi",  
  "name": "Navigate Pro",  
  "version": "18.0.3.0"  
}],
```

## Subscribing Navigate Pro Button on Documents

Being a partner of Navigate Pro (NP) gives you 2 way to launch Navigate pro functionality:

1. Subscribing original NP buttons on documents;
2. Adding your own NP buttons.

Here we introduce a way to realize point 1.

Each NP button raise an event “OnBeforeStartingNavigateProPageXX” (Note see Navigate Pro documentation to get the managed pages. XX is the ID of the page: i.e. 42 is Sales Order Page), with 2 parameters:

1. Rec (source table of the page): the record from which NP is fired;
2. isHandled (Boolean): to avoid original NP routine to fire.

```
[EventSubscriber(ObjectType::Codeunit, 70104170, 'onBeforeStartNavigateProPage42', '', false, false)]
0 references
procedure fStartFromSalesOrder(Rec: record 36; var isHandled:boolean)
//This code is just for example (a working one). You can optimize as you wish.
var
    myJsonObject: JsonObject;
    //record variables to manage
    tSalesHeader: Record 36;

begin
    isHandled := true; //To skip original behaviors

    //Fill json with documents and relationships.
    //-----
    clear(jsonData);
    //-----
    //Here you can add your own search routine to find documents, filling a json object
    fPrepareMyDocForAddin(); //Just for example
    fPrepareMyRelForAddin(); //Just for example
    //-----
    //Send Data to Link codeunit
    cuNavigateProserializer.fSetJsonData(jsonData);
    //-----
    //Initialize addin page
    pageNavigatePro.fPartnerInitializer();
    //Run Addinpage
    pageNavigatePro.run();
end;
```

Example of a “Press Button” routine

The process in the subscribing routine can be divided in 3 main phases:

1. Set *isHandle* flag to prevent NP to fire original code;
2. Preparing the data that you want that the graphical engine shows. I.e. Here you can find your documents;
3. Initialize and run NP addin page.

## Json data structures

To send data to the addin engine, you must provide a json file to the application (from here jsonData).

Inside this file there will be a part, with the information you want to draw on the images, and a part to build the lines between the images. The index of the relationship part must be greater than the document part.

**Note:** An example of how to build this file is in the codeunit *PartnerNavigateProMgt*.

The most important part of the json object is the index, that link together all data.

```

▼ 1:
  sType:      "Document"
  sTableName: "NEW1  "
  sDocumentNo: "Document02"
  iLevel:     "0"
  sDate:      "Hello"

▼ 2:
  sType:      "Document"
  sTableName: "NEW2"
  sDocumentNo: "Document020"
  iLevel:     "1"
  sDate:      "01/01/2021"

▼ 3:
  sType:      "Document"
  sTableName: "NEW3"
  sDocumentNo: "Document023"
  iLevel:     "2"
  sDate:      "750604"
  
```

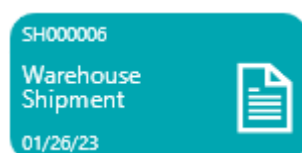
Example of json Data for the images (index is 1,2,3 etc.).

Each image is identified by its index. There are 2 blocks of data inside the json:

- “document”: carry information about what you draw over images. In the original NP, in this part you can see information about documents, but you can draw whatever you want;
- “relationship”: give the information about how drawing lines.

## Document part

In this part of the jsonData you must indicate what the user will see.



Std Document Image of NP

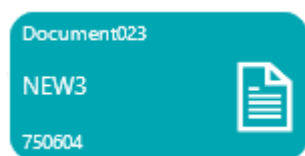
You must fill 5 name-value pairs. All the pairs are mandatory. The name is fixed:

- sType: the value is fixed. It must contain "Document". It is not printed on video;
- sTableName: it contains the string that will be printed in the middle position. In NP standard it shows the page name, but you can send any string you want;
- sDocumentNo: it contains the string that will be printed in the first line. In NP standard it shows the "document no.", but you can send any string you want;
- iLevel: it contains an integer (passed as string), from 0 to 5. It indicates the starting position of the image, from left to right. 0 is the far-left column, while 5 is the last column on the right. It must be always filled with an integer like value.
- sDate: it contains the string that will be printed in the last line. In NP standard it shows a date, but you can send any string you want.

With Version 18.0.3.0 there are 2 more key values, not visible in the flow:

- sDescription: Currently not used. For future internal developments.
- bDocExists: to indicate an existing document. Useful to skip opening documents no more existing (i.e. an invoiced Sales Order).

Following the json of the previous paragraph the image with index 3 will be:



An image filled with random data.

## Relationship Part

In this part of the jsonData you must indicate how to connect the image you drew.

▼ 5:

```
sType: "Relationship"
iFatherEntryNo: "1"
iDocEntryNo: "2"
iLinkType: "0"
```

▼ 6:

```
sType: "Relationship"
iFatherEntryNo: "1"
iDocEntryNo: "3"
iLinkType: "1"
```

▼ 7:

```
sType: "Relationship"
iFatherEntryNo: "3"
iDocEntryNo: "4"
iLinkType: "1"
```

There are 4 name-value pairs to fill. All the pairs are mandatory. The name is fixed:

- sType: the value is fixed. It must contain "Relationship". It is not printed on video;
- iFatherEntryNo: it contains an integer (as string). Each line is drawn between 2 documents. The number here is the index of the jsonData from which you start drawing the line (i.e. 1 in this field

indicate the document having NEW1 as sTablename). Index must be referred to a Document object;

- iDocEntryNo: it contains an integer (as string). The number here must be referred to a Document and indicate the arrow part of the relationship;
- iLinkType: it contains an integer (as string). It must be 0, full line, or 1, dashed line. Putting here different numbers can lead to problems in drawing lines.



The line between document with index 1 and the document with index 2 in the previous json image. The iLinkType is 0, producing a full line.



## Events

---

The program gives the partner the possibility to subscribe 2 events:

```
[IntegrationEvent(false, false)]
local procedure OnRightClickEventEntryNo(iEntryNo: integer);
begin
end;

[IntegrationEvent(false, false)]
local procedure OnDoubleClickEventEntryNoJson(iEntryNo: integer;
                                              jsonData: jsonobject;
                                              bIsHandled: Boolean);

begin

end;
```

They are used to permit the customization of 2 flows, linked on Right Click and Double Click of the mouse. In the Info Studi Navigate Pro program, double click is used to open the documents, while Right Click was open to all for customization.

**TIP: put bIsHandle to true to skip the normal NP flow. If bIsHandle is false the application could open more than one document.**

Partner using this procedure, can be interested in receiving the whole json data information, since the call is completely asynchronous.