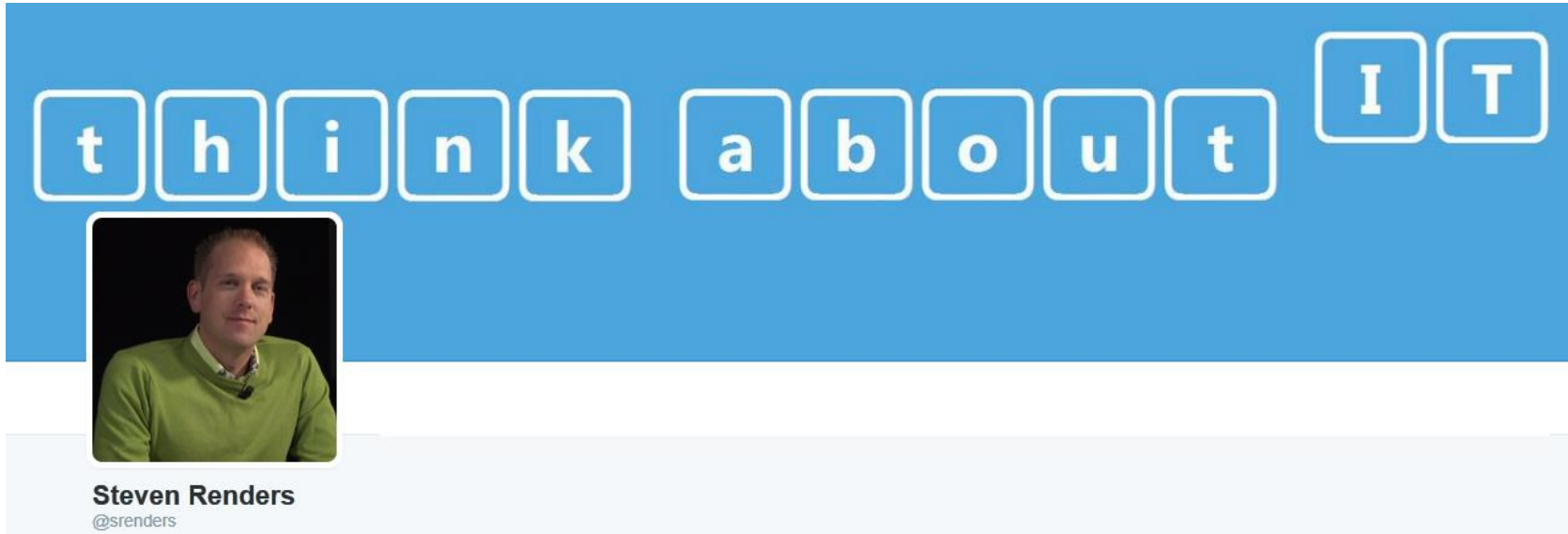
New CAL Functions in Dynamics NAV 2017

NAVSKILLS

LIBERTY GROVE SOFTWARE

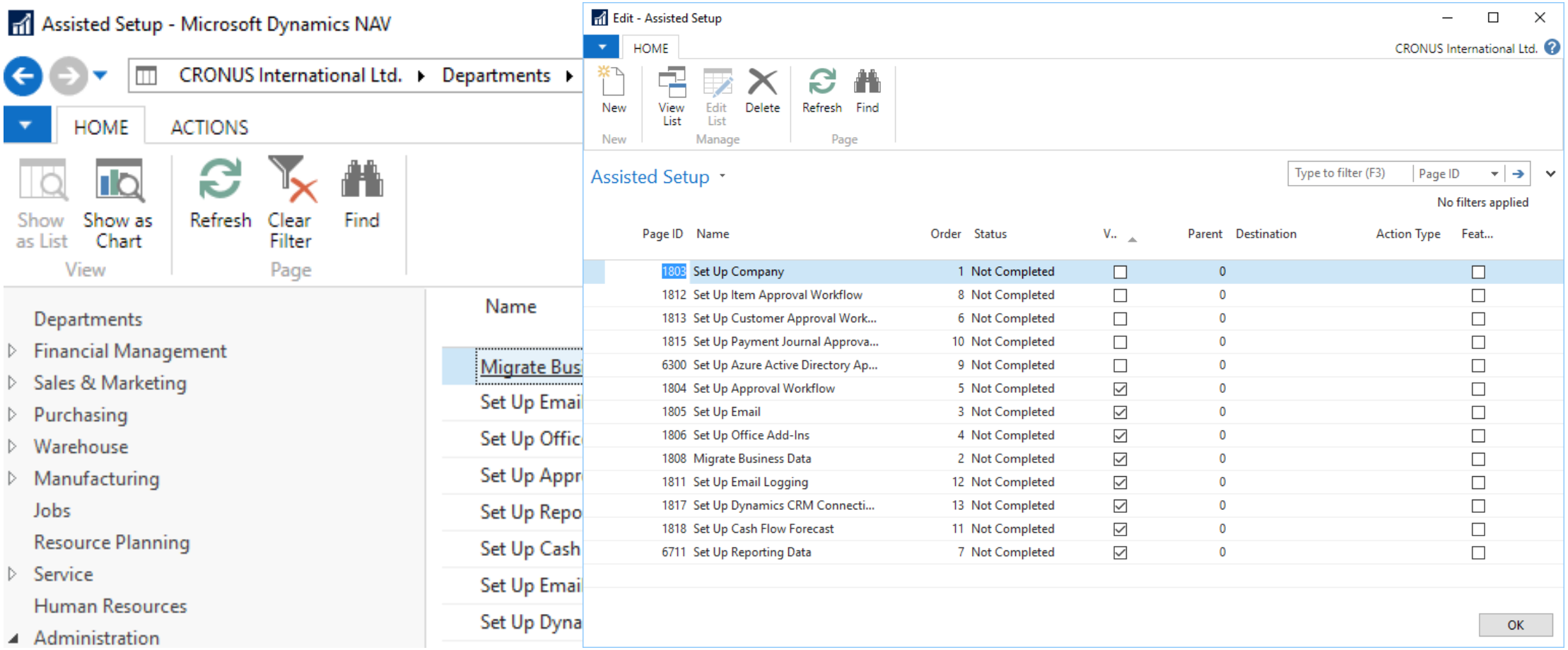


Introduction

- Assisted Setup
- Page Wizard
- Working With Media on Records
- Task Scheduler
- Notifications
- Time Series API
- Resources / Help

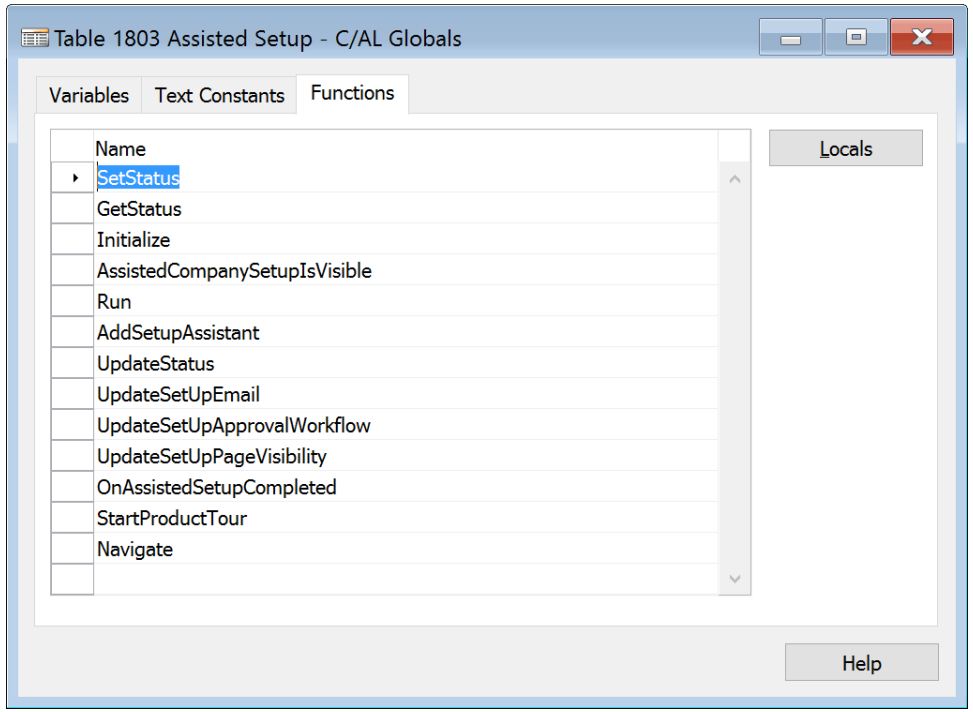
Assisted Setup

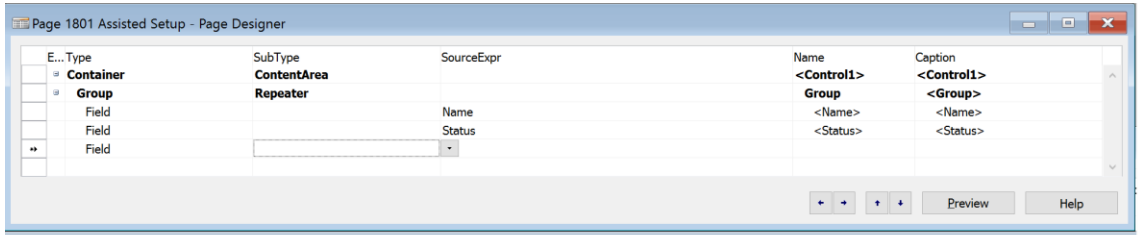
Assisted Setup



Assisted Setup

- How does it work?
  - Table 1803 Assisted Setup





Page 1801 Assisted Setup - C/AL Editor

```
1  Documentation()
2
3  OnInit()
4
5  OnOpenPage()
6   Initialize;
```

Table 1803 Assisted Setup - C/AL Editor

```
63 Initialize()
64 IF NOT ISEMPTY THEN BEGIN
65   UpdateStatus;
66   EXIT;
67 END;
68
69 SortingOrder := 1;
70 AddSetupAssistant(PAGE::"Assisted Company Setup Wizard",InitialCompanySetupTxt,SortingOrder,AssistedCompanySet
71 AddSetupAssistant(PAGE::"Data Migration Wizard",DataMigrationTxt,SortingOrder,TRUE);
72 AddSetupAssistant(PAGE::"Email Setup Wizard",SMTPSetupTxt,SortingOrder,TRUE);
73 AddSetupAssistant(PAGE::"Exchange Setup Wizard",OfficeAddinSetupTxt,SortingOrder,TRUE);
74 AddSetupAssistant(PAGE::"Approval Workflow Setup Wizard",ApprovalWorkflowSetupTxt,SortingOrder,TRUE);
75 AddSetupAssistant(PAGE::"Cust. Approval WF Setup Wizard",CustomerAppWorkflowTxt,SortingOrder,FALSE);
76 AddSetupAssistant(PAGE::"OData Setup Wizard",ODataWizardTxt,SortingOrder,TRUE);
77 AddSetupAssistant(PAGE::"Item Approval WF Setup Wizard",ItemAppWorkflowTxt,SortingOrder,FALSE);
78 AddSetupAssistant(PAGE::"Azure AD App Setup Wizard",AzureAdSetupTxt,SortingOrder,FALSE);
79 AddSetupAssistant(PAGE::"Pmt. App. Workflow Setup Wzrd.",PmtJnlAppWorkflowTxt,SortingOrder,FALSE);
80 AddSetupAssistant(PAGE::"Cash Flow Forecast Wizard",CashFlowForecastTxt,SortingOrder,TRUE);
81 AddSetupAssistant(PAGE::"Setup Email Logging",SetupEmailLoggingTxt,SortingOrder,TRUE);
82 AddSetupAssistant(PAGE::"CRM Connection Setup Wizard",CRMConnectionSetupTxt,SortingOrder,TRUE);
```

Assisted Setup

Table 1803 Assiste

```
89  Run(
90    IF S
91      CA
92
93
94
95
96
97      EN
98    IF D
99      Te
100   ELSE
101     EV
102   PAGE
```

Edit - Assisted Setup

HOME

CRONUS International Ltd.

New | View List | Edit List | Delete | Refresh | Find

New | Manage | Page

Assisted Setup

Page ID	Name
1803	et Up Company
1804	et Up Approval Workflow
1805	et Up Email
1806	et Up Office Add-Ins
1808	igrate Business Data
1811	et Up Email Logging
1812	et Up Item Approval Workflow
1813	et Up Customer Approval Work...
1815	et Up Payment Journal Approva...
1817	et Up Dynamics CRM Connecti...
1818	et Up Cash Flow Forecast
6300	et Up Azure Active Directory Ap...
6711	et Up Reporting Data

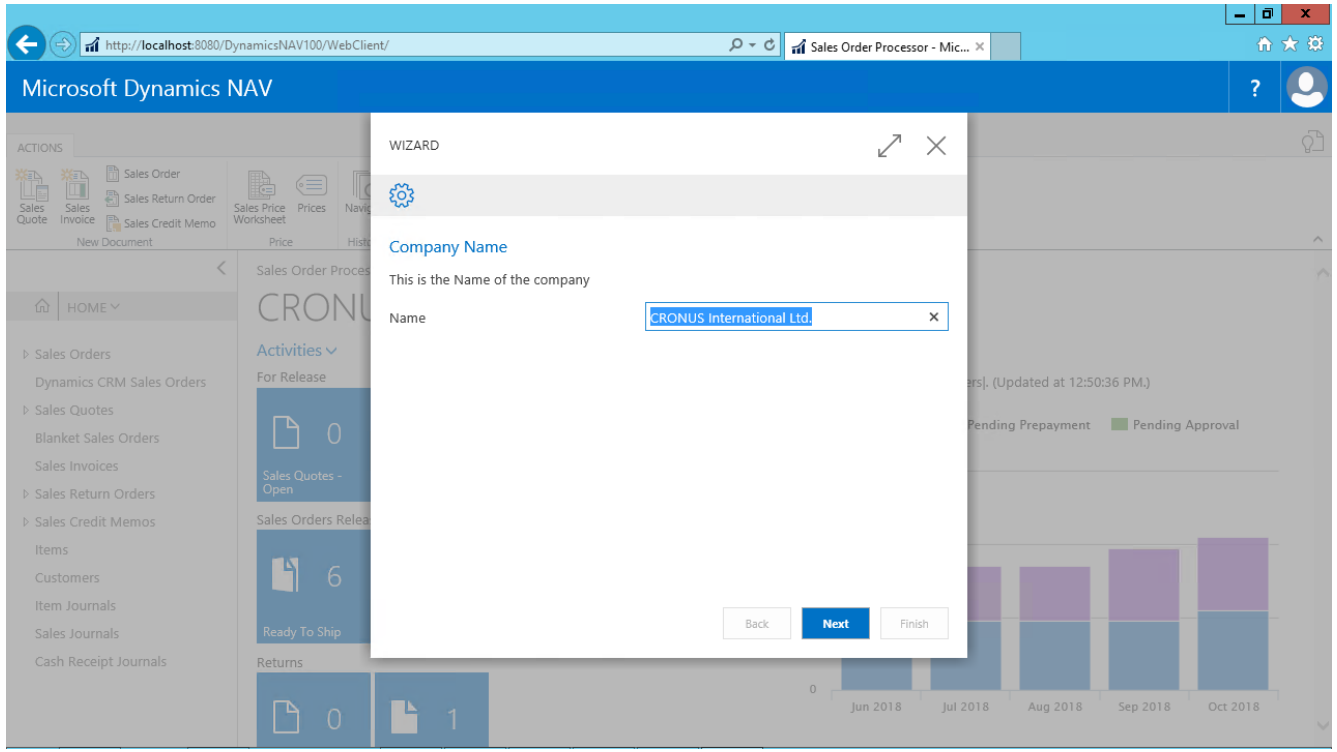
Table
Page
Report
Codeunit
Query
XMLport
MenuSuite
All

Type	ID	Name	Modified	Version List
	1801	Assisted Setup		NAVW110.0
	1802	Assisted Setup Part		NAVW110.0
	1803	Assisted Company Setup Wizard		NAVW110.0
	1804	Approval Workflow Setup Wizard		NAVW110.0
	1805	Email Setup Wizard		NAVW110.0
	1806	Exchange Setup Wizard		NAVW110.0
	1807	Data Migration Settings		NAVW110.0
	1808	Data Migration Wizard		NAVW110.0
	1809	Data Migrators		NAVW110.0
	1810	Data Migration Entities		NAVW110.0
	1811	Setup Email Logging		NAVW110.0
	1812	Item Approval WF Setup Wizard		NAVW110.0
	1813	Cust. Approval WF Setup Wizard		NAVW110.0
	1815	Pmt. App. Workflow Setup Wzrd.		NAVW110.0
	1816	Job Creation Wizard		NAVW110.0
	1817	CRM Connection Setup Wizard		NAVW110.0
	1818	Cash Flow Forecast Wizard		NAVW110.0

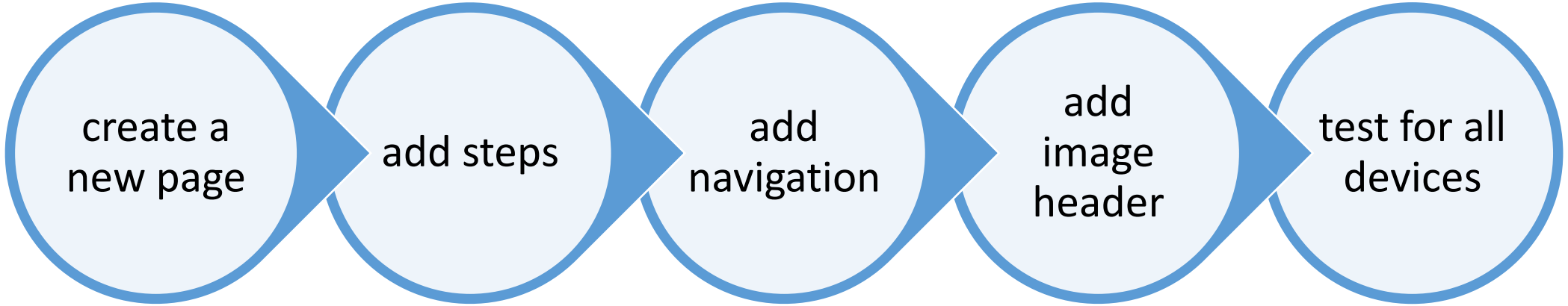
Page Wizard

Page Wizard

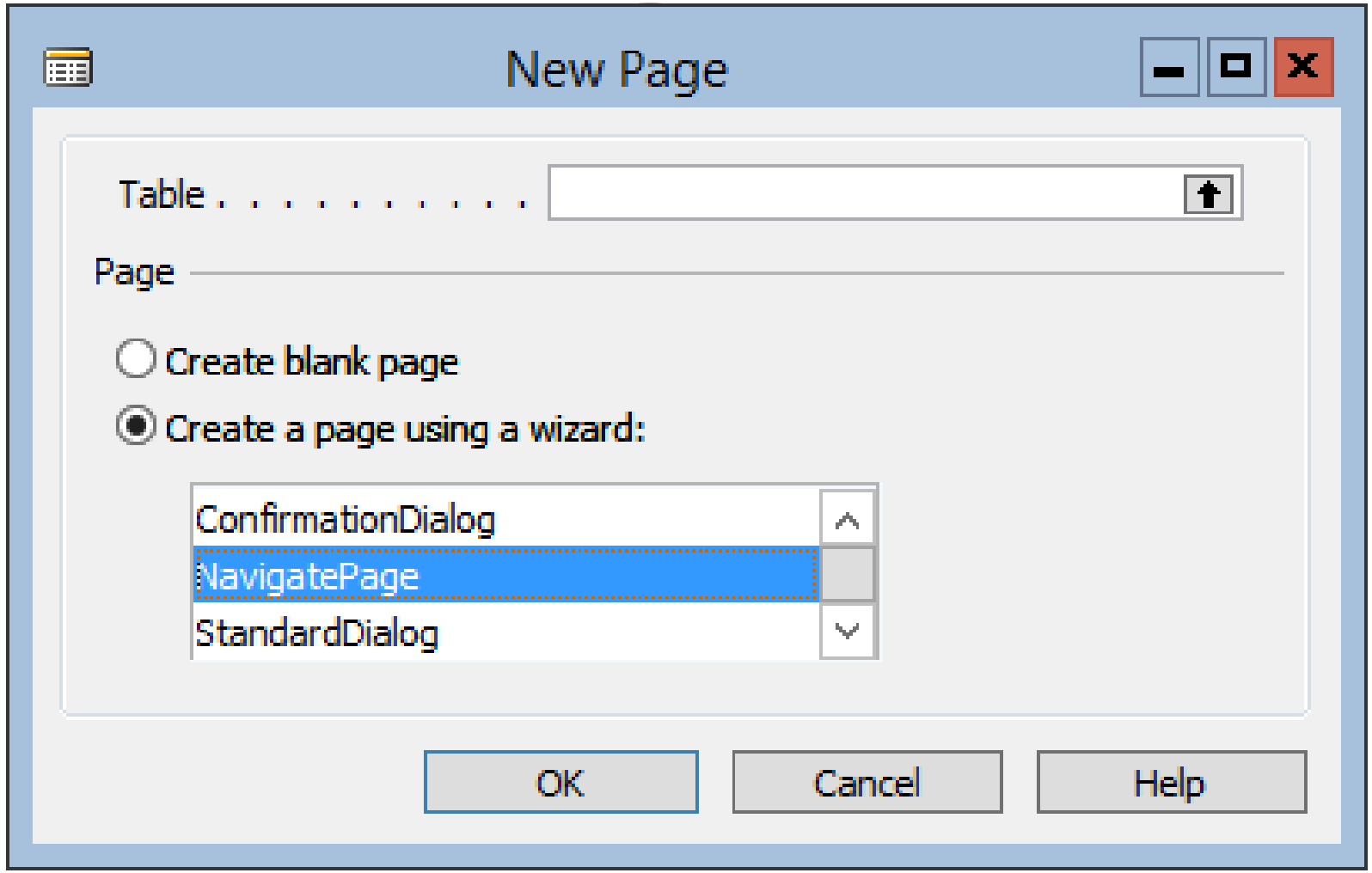
- How to create a Wizard page?



Page Wizard


create a new page
add steps
add navigation
add image header
test for all devices

Page Wizard


New Page
Table . . . . . .
Page
Create blank page
Create a page using a wizard:
ConfirmationDialog
NavigatePage
StandardDialog
OK
Cancel
Help

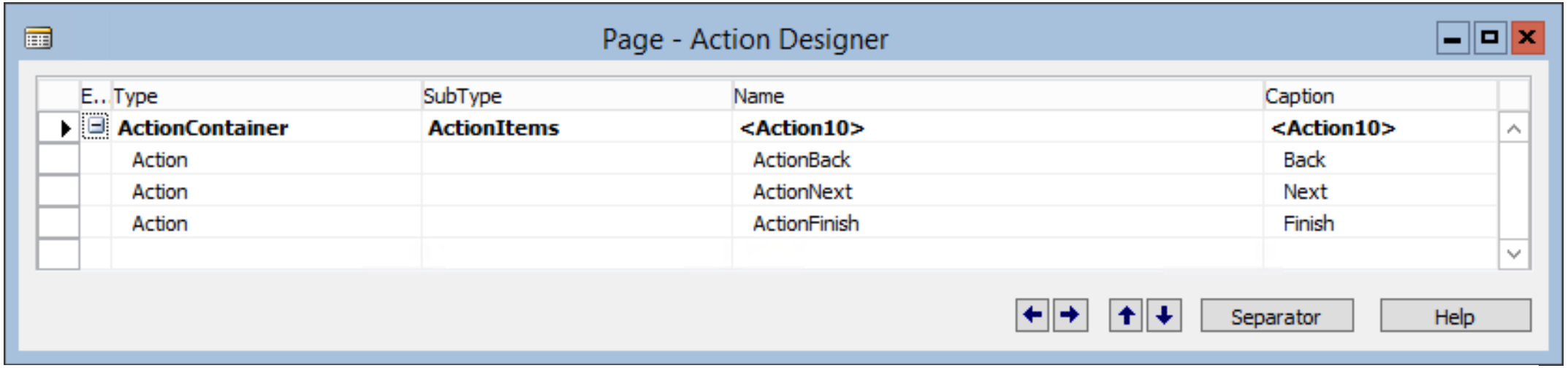
Page Wizard

Page 70000 Wizard - Page Designer

E...	Type	SubType	SourceExpr	Name	Caption
⊟	**Container**	**ContentArea**		**<Control1>**	**<Control1>**
⊞	***Group***	***Group***		***<Control17>***	***<Control17>***
⊞	***Group***	***Group***		***<Control15>***	***<Control15>***
⊟	**Group**	**Group**		**Step1**	**<Step1>**
⊟	**Group**	**Group**		**Company Name**	**<Company Name>**
	Field		Name	<Name>	<Name>
⊟	**Group**	**Group**		**Step2**	**<Step2>**
⊟	**Group**	**Group**		**Email Address**	**<Email Address>**
	Field		"E-Mail"	<E-Mail>	<Email>
⊟	**Group**	**Group**		**Step3**	**<Step3>**
⊟	**Group**	**Group**		**All Done!**	**<All Done!>**

Preview Help

Page Wizard

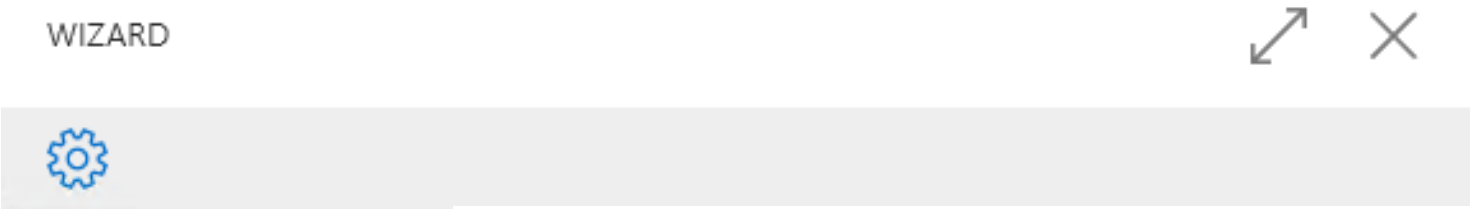

Page - Action Designer
E..
Type
SubType
Name
Caption
ActionContainer
ActionItems
<Action10>
<Action10>
Action
ActionBack
Back
Action
ActionNext
Next
Action
ActionFinish
Finish
Separator
Help

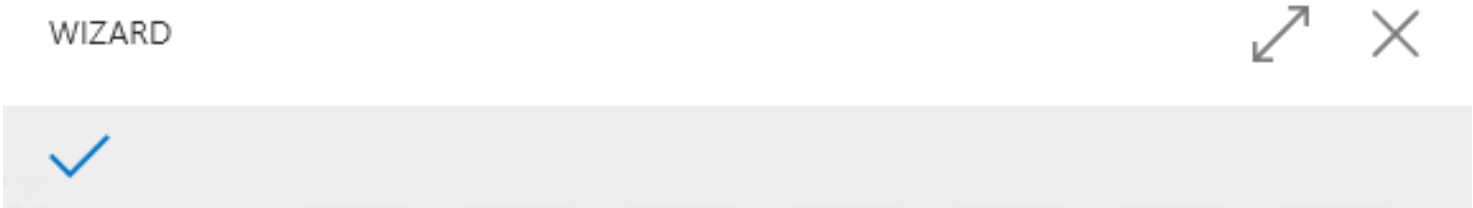

Back
Next
Finish

Page Wizard

Page 70000 Wizard - Page Designer

E...	Type	SubType	SourceExpr	Name	Caption
	Container	**ContentArea**		**<Control1>**	**<Control1>**
	Group	**Group**		**<Control17>**	**<Control17>**
	Field		MediaRepositoryStandard.Image	<MediaRepositoryStandard>	<MediaRepositoryStandard>
	Group	**Group**		**<Control15>**	**<Control15>**
	Field		MediaRepositoryDone.Image	<MediaRepositoryDone>	<MediaRepositoryDone>
	Group	**Group**		**Step1**	**<Step1>**





Page Wizard

Test for all devices

Test the Phone, Tablet and Web clients

- http://localhost:8080/DynamicsNAV100/WebClient/
- http://localhost:8080/DynamicsNAV100/WebClient/tablet.aspx
- http://localhost:8080/DynamicsNAV100/WebClient/phone.aspx

Working With Media on Records

Working With Media on Records

You can upload media files

- such as images, to the database
- for displaying with records in the client.

Use a **BLOB** data type

- You add the media file to a BLOB data type field.

Use a **Media** or **MediaSet** data type

- enables you to store media objects (images) in **system** tables
- and then **reference** the images from application **records**.

Working With Media on Records

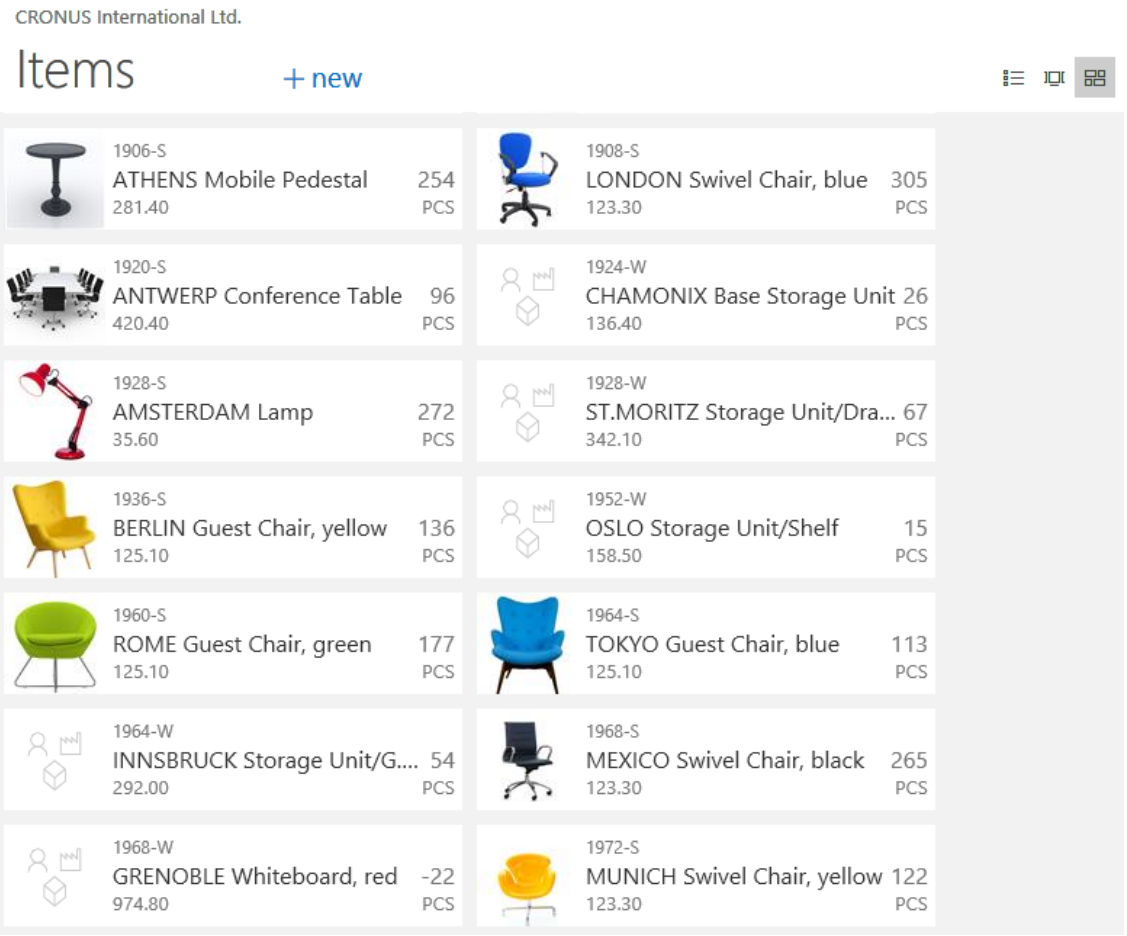
- For **Web** client and **Universal App**
  - use this method to display **images** w
  - when the page is viewed in the **Brick**

Table 27 Item - Table Designer

E..	Field No.	Field Name	Data Type	Length	Descriptio
✔	91	Gen. Prod. Posting Group	Code	10	
✔	92	Picture	MediaSet		

Table 27 Item - Field Groups

ID	Name	Group
1	DropDown	No.,Description,Base Unit of Measure,Unit Price
2	Brick	No.,Description,Inventory,Unit Price,Base Unit of Measure,Descriptio



Working With Media on Records

- Using the **Media** or **MediaSet** data type
  - provides **better performance** than using a BLOB data type
  - and is **more flexible** in its design.
- With a **BLOB** data type
  - each time the media is rendered in the client, it is retrieved from the SQL database server,
  - which requires extra **bandwidth** and affects performance.
- With the **Media** and **MediaSet** data types
  - the client media ID to **cache** the media data,
  - improves the response time for rendering the media.

Working With Media on Records

- Using Media and Media Sets on Records:
  - Table fields support two data types for media: **Media** and **MediaSet**.
- **Media** associates a record with a **single media file**.
  - For example, to display an image with each record in a list type page.
- **MediaSet** associates a record with **one or more** media files
  - which lets you set up a **collection** or **catalogue** of media for a record.
    - use this function to set up a **slide show** of images in a **card** type page.

Working With Media on Records

- You can add media to the record
  - either from a **file** or passed in an **InStream** object.
- Media is imported and then stored as objects
  - in the system table **2000000184 Tenant Media**
  - of the application database
  - and each media object is assigned a unique identifier
    - (**ID**).

Working With Media on Records

- If a media is added to a **Media** data type field
  - the field references the media by its **ID**.
- If the media is added to **MediaSet** data type field
  - the media file is assigned to a media set in system table 2000000183 **Tenant Media Set**.
  - has a **unique identifier**, which is referenced from the field.
  - is **created** with the **first file** that you add on the record.
  - Any **additional** media files for the record are then **associated** with the same **media set**.

Working With Media on Records

- Supported Media Types:
  - The media type, sometimes referred to as the **MIME** type, is an **Internet standard** to describe the **contents** of a **file**.
  - Currently, Microsoft Dynamics NAV only supports **image** type
    - more specifically, only those image subtypes that are supported by the **System.Drawing.Image** class of the .NET Framework.

Working With Media on Records

Obtain the media file or files that you want to use on record.

Modify the **table** object to include a **field** that has the data type **Media** or **MediaSet**.

Add **C/AL code** that **imports** the media on the field.

The Media and MediaSet data types support several C/AL **functions**.

Working With Media on Records

- **Media**
  - **IMPORTFILE** Function (Media)
    - Sets up an image on a record from a file.
    - The media file is imported to the application database.
  - **IMPORTSTREAM** Function (Media)
    - Sets up an image on a record from an InStream object.
    - The media file is imported to the application database.
  - **HASVALUE** Function (Media)
    - Detects whether a record has a media object in the Media data type field.
  - **MEDIAID** Function (Media)
    - Gets the unique identifier (GUID) that is assigned to the media object in the application database.
  - **EXPORTFILE** Function (Media)
    - Exports a media object that is set up on a record to a file.
  - **EXPORTSTREAM** Function (Media)
    - Exports a media object that is set up on a record to an OutStream object.

Working With Media on Records

Page 346 Item Picture - C/AL Editor

```
1   Documentation()
2
3   TakePicture - OnAction()...
10  ImportPicture - OnAction()
11  IF Picture.COUNT > 0 THEN
12    IF NOT CONFIRM(OverrideImageQst) THEN
13      EXIT;
14
15  FileName := FileManagement.UploadFile(SelectPictureTxt,ClientFileName);
16  IF FileName = '' THEN                                  ClientFileName : Text
17    EXIT;
18
19  CLEAR(Picture);
20  Picture.IMPORTFILE(FileName,ClientFileName);
21  MODIFY(TRUE);
22  IF FileManagement.DeleteServerFile(FileName) THEN;
23
24  ExportFile - OnAction()...
35  DeletePicture - OnAction()...
42  LOCAL SetEditableOnPictureActions()
43  DeleteExportEnabled := Picture.COUNT <> 0;
44
45  CameraProvider::PictureAvailable(PictureName : Text;PictureFilePath : Text)...
```

100 %

109	Res. Qty. on Sales Returns	Decimal
110	Res. Qty. on Purch. Returns	Decimal
120	Stockout Warning	Option

Extended Texts | Translations | Dimensions | Attributes | Item
Ledger Entries | Phys. Inventory Ledger Entries | History
Set Special Prices | Set Special Discounts | Special Prices & Discounts Overview | Special Prices & Discounts
Send Approval Request | Cancel Approval Request | Request Approval
Notes | Links | Show Attached
Previous | Next | Page

Show more

Picture

ner

	SubType	Caption	Name
ner	ActionItems	<Action2>	<Action2>
		Take	TakePicture
		Import	ImportPicture
		Export	ExportFile
		Delete	DeletePicture

Separator | Help

	SubType	SourceExpr	Name	Caption
	ContentArea		<Control190000...	<Control1900000001>
		ture	<Picture>	<Picture>

Preview | Help

Working With Media on Records

Tenant Media Thumbnails

Type to filter (F3) | ID

ID	Media ID	C...	Mime Type	Height	Width	Company Name	Em...	Height	Width	Company Name
{203bf6ff-ea5e-4543-be1b-0250ece2d62e}	{9701a061-7ca6-4dd8-bf24-78759b651ef3}		image/jpeg	360	360	CRONUS Internatio...	☐	375	500	CRONUS International Ltd.
{7cd09e7b-d352-425c-9da6-0372325fa5e7}	{874a9cb9-5d46-4940-a3f5-11ea9325cb9c}		image/jpeg	270	360	CRONUS Internatio...	☐	500	400	CRONUS International Ltd.
{deeb42dc-794f-4388-9c29-06160560a7c0}	{8f678d0f-fc41-4dd8-b7ed-bf67931103cb}		image/jpeg	240	240	CRONUS Internatio...	☐	600	500	CRONUS International Ltd.
{1d2ba1e9-9046-4911-abc3-0693b60977e2}	{383e0ffe-1b84-42d6-8478-aece04bcc081}		image/jpeg	450	360	CRONUS Internatio...	☐	500	410	CRONUS International Ltd.
{0bc7427f-288f-4bfc-b4b4-06fc84ca1672}	{74a91d5a-efc3-41bb-b55e-ef200a9be6a9}		image/jpeg	241	360	CRONUS Internatio...	☐	400	400	
{9f6639dd-98a0-4afb-b798-07ce2359caed}	{6c74f2fa-7eac-432e-a2b0-2053dd09a557}		image/jpeg	70	70	CRONUS Internatio...	☐	400	1000	
{b8e2238b-8d8b-4ea3-a6a1-07fd9a9abf76}	{36394197-59f1-4e89-ac4f-fd37881144a6}		image/jpeg	288	360	CRONUS Internatio...	☐	466	500	CRONUS International Ltd.
{5304b0af-60d1-4c67-8632-088ca6a87ef0}	{874a9cb9-5d46-4940-a3f5-11ea9325cb9c}		image/jpeg	70	70	CRONUS Internatio...	☐	500	482	CRONUS International Ltd.
{8d8d6caa-cc44-450f-a54e-0ade2a6a30e5}	{8f678d0f-fc41-4dd8-b7ed-bf67931103cb}		image/jpeg	70	70	CRONUS Internatio...	☐	700	1000	
{3f0691a7-5dbe-4d52-8d85-19afdffc4eba}	{644a3bf7-4b00-4c2b-b8fe-526f09b024b2}		image/jpeg	432	360	CRONUS Internatio...	☐	400	1000	
{6c82b753-8bd2-4221-b1ad-1b047541c08e}	{36394197-59f1-4e89-ac4f-fd37881144a6}		image/jpeg	240	240	CRONUS Internatio...	☐	496	500	CRONUS International Ltd.
{e34c61b9-e03e-449e-b4f4-1cfc7236d08b}	{644a3bf7-4b00-4c2b-b8fe-526f09b024b2}		image/jpeg	70	70	CRONUS Internatio...	☐	500	410	CRONUS International Ltd.
{79bb43dd-23e7-4e0a-a341-21b67e18bc05}	{b59e914f-d335-4270-8a28-8796b6d8cfb5}		image/png	29	362		☐	335	500	CRONUS International Ltd.
{582b8169-f640-4cc3-9327-21bd45e606f0}	{8c94b4e5-589c-4ec0-85e3-af7371e2946f}		image/jpeg	432	360	CRONUS Internatio...	☐	400	1000	
{5461f296-a000-44a9-8920-229d9b24cdd1}	{d80f0127-3aa4-4ef0-b918-d33aa78897e2}		image/png	240	240		☐	496	500	CRONUS International Ltd.
{6a0b83a9-85b3-4207-94a4-269d62d24e04}	{af397c38-bffb-41cf-83eb-fb32b4a787d9}		image/jpeg	70	70	CRONUS Internatio...	☐	400	500	CRONUS International Ltd.
{954d720c-9f24-43c1-9f89-2a9e42aa7950}	{b600780f-820f-474b-8ff7-aa1a1a92b829}		image/jpeg	70	70	CRONUS Internatio...	☐			
{40bbc091-23f3-4b95-9c74-2d21fb06087d}	{b600780f-820f-474b-8ff7-aa1a1a92b829}		image/jpeg	240	240	CRONUS Internatio...	☐			

Working With Media on Records

- **MediaSet**
  - **IMPORTFILE** Function (MediaSet)
    - Sets up an image on a record from a file and assigns the image to a media set.
    - The media file is imported to the application database.
  - **IMPORTSTREAM** Function (MediaSet)
    - Sets up an image on a record from an InStream object and assigns it to a media set.
    - The media file is imported to the application database.
  - **MEDIAID** Function (MediaSet)
    - Gets the unique identifier (GUID) that is assigned to the media set on a record.
  - **COUNT** Function (MediaSet)
    - Gets the total number of media objects that are included in the media set on a record.
  - **EXPORTFILE** Function
    - Exports the media objects that included in a media set to individual files.

Working With Media on Records

Table 9400 Media Repository - Table Designer

E..	Field No.	Field Name	Data Type	Length	Description
✔	1	File Name	Text	250	
✔	2	Display Target	Code	50	
✔	3	Image	Media		
✔	4				

Help

Table 9400 Media Repository - C/AL Editor

```
22
23 ImportMedia(FilePath : Text;DisplayTarget : Code[50])
24 IF FileManagement.ServerFileExists(FilePath) THEN BEGIN
25   FileName := COPYSTR(FileManagement.GetFileName(FilePath),1,MAXSTRLEN(FileName));
26   IF NOT GET(FileName,DisplayTarget) THEN BEGIN
27     INIT;
28     "File Name" := FileName;
29     "Display Target" := DisplayTarget;
30     INSERT(TRUE);
31   END;
32   Image.IMPORTFILE(FilePath,FileName);
33   MODIFY(TRUE);
34 END ELSE
35   ERROR(FileDoesNotExistErr,FilePath);
36
```

100 %

Used By

Page (18)

1803 Assisted Company Setup Wizard (v)
1804 Approval Workflow Setup Wizard (v)
1805 Email Setup Wizard (v)
1806 Exchange Setup Wizard (v)
1808 Data Migration Wizard (v)
1811 Setup Email Logging (v)
1812 Item Approval WF Setup Wizard (v)
1813 Cust. Approval WF Setup Wizard (v)
1815 Pmt. App. Workflow Setup Wzrd. (v)
1816 Job Creation Wizard (v)
1817 CRM Connection Setup Wizard (v)
1818 Cash Flow Forecast Wizard (v)
1824 Jobs Setup Wizard (v)
1826 QuickBooks Setup Wizard (v)
5551 Fixed Asset Acquisition Wizard (v)
6300 Azure AD App Setup Wizard (v)
6711 OData Setup Wizard (v)
9192 Company Creation Wizard (v)

Working With Media on Records

Page 1805 Email Setup Wizard - Page Designer

E..	Type	SubType	SourceExpr	Name	Caption
	Container	ContentArea		<Control1>	<Control1>
	Group	Group		<Control96>	<Control96>
	Field		MediaRepositoryStandard.Image	<MediaReposit...	<MediaRepositoryStandard>
	Group	Group		<Control98>	<Control98>
	Field		MediaRepositoryDone.Image	<MediaReposit...	<MediaRepositoryDone>
	Group	Group		<Control20>	<Control20>
	Group	Group		<Control2>	<Control2>

Page 1805 Email Setup Wizard - C/AL Editor

```
LOCAL LoadTopBanners()
IF MediaRepositoryStandard.GET('AssistedSetup-NoText-400px.png',FORMAT(CURRENTCLIENTTYPE)) AND
   MediaRepositoryDone.GET('AssistedSetupDone-NoText-400px.png',FORMAT(CURRENTCLIENTTYPE))
THEN
  TopBannerVisible := MediaRepositoryDone.Image.HASVALUE;
```

100 %

Working With Media on Records

Codeunit 8611 Config. Package Management - C/AL Editor

```
1379 IsMediaSetField(TableId : Integer;FieldId : Integer) : Boolean
1380 Field.SETRANGE(TableNo,TableId);
1381 Field.SETRANGE("No.",FieldId);
1382 IF Field.FINDFIRST THEN
1383   EXIT(Field.Type = Field.Type::MediaSet);
1384
1385 EXIT(FALSE);
1386
1387 LOCAL ImportMediaSetFiles(VAR ConfigPackageData : Record "Config. Package Da
1388 IF NOT CanImportMediaField(ConfigPackageData,FieldRef,DoModify,MediaSetID) T
1389   EXIT;
1390
1391 MediaSetIDConfigPackageData.SETRANGE("Package Code",ConfigPackageData."Packa
1392 MediaSetIDConfigPackageData.SETRANGE("Table ID",DATABASE::"Config. Media Buf
1393 MediaSetIDConfigPackageData.SETRANGE("Field ID",TempConfigMediaBuffer.FIELDN
1394 MediaSetIDConfigPackageData.SETRANGE(Value,MediaSetID);
1395
1396 IF NOT MediaSetIDConfigPackageData.FINDSET THEN
1397   EXIT;
1398
1399 TempConfigMediaBuffer.INIT;
1400 TempConfigMediaBuffer.INSERT;
1401 BlobMediaSetConfigPackageData.SETAUTOCALCFIELDS("BLOB Value");
1402
1403 REPEAT
1404   BlobMediaSetConfigPackageData.GET(
1405     MediaSetIDConfigPackageData."Package Code",MediaSetIDConfigPackageData."
1406     TempConfigMediaBuffer.FIELDNO("Media Blob"));
1407   BlobMediaSetConfigPackageData."BLOB Value".CREATEINSTREAM(BlobInStream);
1408   TempConfigMediaBuffer."Media Set".IMPORTSTREAM(BlobInStream,'');
1409   TempConfigMediaBuffer.MODIFY;
1410 UNTIL MediaSetIDConfigPackageData.NEXT = 0;
1411
1412 FieldRef.VALUE := FORMAT(TempConfigMediaBuffer."Media Set");
1413
```

Codeunit 8611 Config. Package Management - C/AL Editor

```
1414 IsMediaField(TableId : Integer;FieldId : Integer) : Boolean
1415 Field.SETRANGE(TableNo,TableId);
1416 Field.SETRANGE("No.",FieldId);
1417 IF Field.FINDFIRST THEN
1418   EXIT(Field.Type = Field.Type::Media);
1419
1420 EXIT(FALSE);
1421
1422 LOCAL ImportMediaFiles(VAR ConfigPackageData : Record "Config. Package Data";VAR FieldRef : FieldRef;DoModify : Boolean)
1423 IF NOT CanImportMediaField(ConfigPackageData,FieldRef,DoModify,MediaID) THEN
1424   EXIT;
1425
1426 MediaIDConfigPackageData.SETRANGE("Package Code",ConfigPackageData."Package Code");
1427 MediaIDConfigPackageData.SETRANGE("Table ID",DATABASE::"Config. Media Buffer");
1428 MediaIDConfigPackageData.SETRANGE("Field ID",TempConfigMediaBuffer.FIELDNO("Media ID"));
1429 MediaIDConfigPackageData.SETRANGE(Value,MediaID);
1430
1431 IF NOT MediaIDConfigPackageData.FINDFIRST THEN
1432   EXIT;
1433
1434 BlobMediaConfigPackageData.SETAUTOCALCFIELDS("BLOB Value");
1435
1436 BlobMediaConfigPackageData.GET(
1437   MediaIDConfigPackageData."Package Code",MediaIDConfigPackageData."Table ID",MediaIDConfigPackageData."No.",
1438   TempConfigMediaBuffer.FIELDNO("Media Blob"));
1439 BlobMediaConfigPackageData."BLOB Value".CREATEINSTREAM(BlobInStream);
1440
1441 TempConfigMediaBuffer.INIT;
1442 TempConfigMediaBuffer.Media.IMPORTSTREAM(BlobInStream,'');
1443 TempConfigMediaBuffer.INSERT;
1444
1445 FieldRef.VALUE := FORMAT(TempConfigMediaBuffer.Media);
1446
1447 LOCAL CanImportMediaField(VAR ConfigPackageData : Record "Config. Package Data";VAR FieldRef : FieldRef;DoModify : Boolean;VAR MediaID : Text) : Boolean
1448 IF NOT DoModify THEN
1449   EXIT(FALSE);
1450
1451 RecRef := FieldRef.RECORD;
1452 IF RecRef.NUMBER = DATABASE::"Config. Media Buffer" THEN
1453   EXIT(FALSE);
1454
1455 MediaID := FORMAT(ConfigPackageData.Value);
1456 IF (MediaID = FORMAT(DummyNotInitializedGuid)) OR (MediaID = '') THEN
1457   EXIT(FALSE);
1458
1459 EXIT(TRUE);
1460
```

Working With Media on Records

- DEMO

Codeunit 50005 Import Item Pictures - C/AL Editor

```
Documentation()

OnRun()
//ImportPictures;
ImportPicturesWithStream;
MESSAGE('Ready');

LOCAL ImportPictures()
IF Item.FINDFIRST() THEN
BEGIN
  REPEAT
    FileName := 'C:\temp\images\' + FORMAT(Item."No.") + '.jpg';
    IF FILE.EXISTS(FileName) THEN BEGIN
      Item.ItemPicture.IMPORTFILE(FileName, 'Demo image for item ' + FORMAT(Item."No."));
      Item.MODIFY;
    END;
  UNTIL Item.NEXT < 1;
END;

LOCAL ImportPicturesWithStream()
IF Item.FINDFIRST() THEN
BEGIN
  REPEAT
    FileName := 'C:\temp\images\' + FORMAT(Item."No.") + '.jpg';

    IF FILE.EXISTS(FileName) THEN BEGIN
        ImportFile.OPEN(FileName);
        ImportFile.CREATEINSTREAM(InStream);
        Item.ItemPicture.IMPORTSTREAM(InStream, 'Demo image for item ' + FORMAT(Item."No."));
      Item.MODIFY;
      ImportFile.CLOSE;
    END;
  UNTIL Item.NEXT < 1;
END;
```

100 %

Task Scheduler

Task Scheduler

- Task Scheduler is the new platform **engine** that runs **Jobs Queues**.
- **Before**
  - all of the Job Queues were handled in AL and were either started through **NAS** or directly from the **client**.
  - AL approach had **loopy logic** for each Job Queue awaiting new jobs to arrive, keeping **sessions open** for a long time.

Task Scheduler

- The **problem** was
  - when something out of AL expected behavior failed, like a network outage making a SQL error or similar
  - then the error would **shut down** the queue
  - requiring it to be **restarted**.

Task Scheduler

- The new implementation supports
  - **restart** of jobs
  - **tolerance** for transient errors
  - amount of **parallelism**
    - including distribution between multiple NST and more.

Task Scheduler

- Or simply continue to use Job Queue entries in Madeira
  - since the Job Queues now runs on the new engine.
- The old **NAS** is only there
  - if people upgrade an application
  - and still need the feature
  - **until** they have **upgraded** the application.

Task Scheduler

- a new system table named "**Scheduled Task**"
- a new set of **C/AL commands** like
  - TASKSCHEDULER.CREATETASK,
  - TASKSCHEDULER.CANCELTASK, etc.
  - basically insert/delete entries from the "Scheduled Task" table.

Task Scheduler

- The trick is, that the system now knows
  - **who** created the task
  - and then also knows that the task should be run with that user's **permissions**.
- The existing **job queue** functionality has been modified
  - so it runs on top of this new mechanism
  - and consequently, it doesn't need the **Job Queue table** for anything.

Task Scheduler

- **Categories** are still **supported**
  - the execution mechanism will make sure that only one job with a certain category is executed at the same time.
- **Priorities** are **not** supported for now
  - it's **first-in-first-out**.
- In the server configuration you will notice that a new **tab** has arrived:
  - **TaskScheduler** that allows you to **turn** the feature **on** or **off** for a specific **NST** and how many **concurrent** tasks can be executed.
  - What this means is that you can now configure one NST to serve **users** and another one for **background jobs**.

Task Scheduler

- And more:
  - The **default** configuration is that the feature is **turned on**
    - you don't need to configure any job queues to make background jobs work.
    - Also, it doesn't keep any threads alive on the server when there is nothing to do.
    - In case of **server restart**
      - the **background execution** will just **resume automatically**
      - after a few minutes
      - as the NST's may need to synchronize
        - and figure out who should run an aborted job.

Task Scheduler

- And more:
  - Use **categories** for **serializing** certain jobs, e.g. posting jobs to avoid lock timeouts (and deadlocks).
    - When a job is started, it checks if it has a category code.
    - If so, it checks if there is any other job running right now with the same category.
    - If so, the job will reschedule itself for some seconds later.

Task Scheduler

- The task scheduler enables you to **control** when certain operations or processes (in other words tasks) are **run**.
  - A task is a **codeunit** or **report** that is **scheduled** to run at a date a time.
- Tasks run in a **background** session between the Dynamics NAV Server instance and database.
  - Behind the scenes, the task scheduler is used by the **job queue** to process job queue entries that are created and managed from the clients.

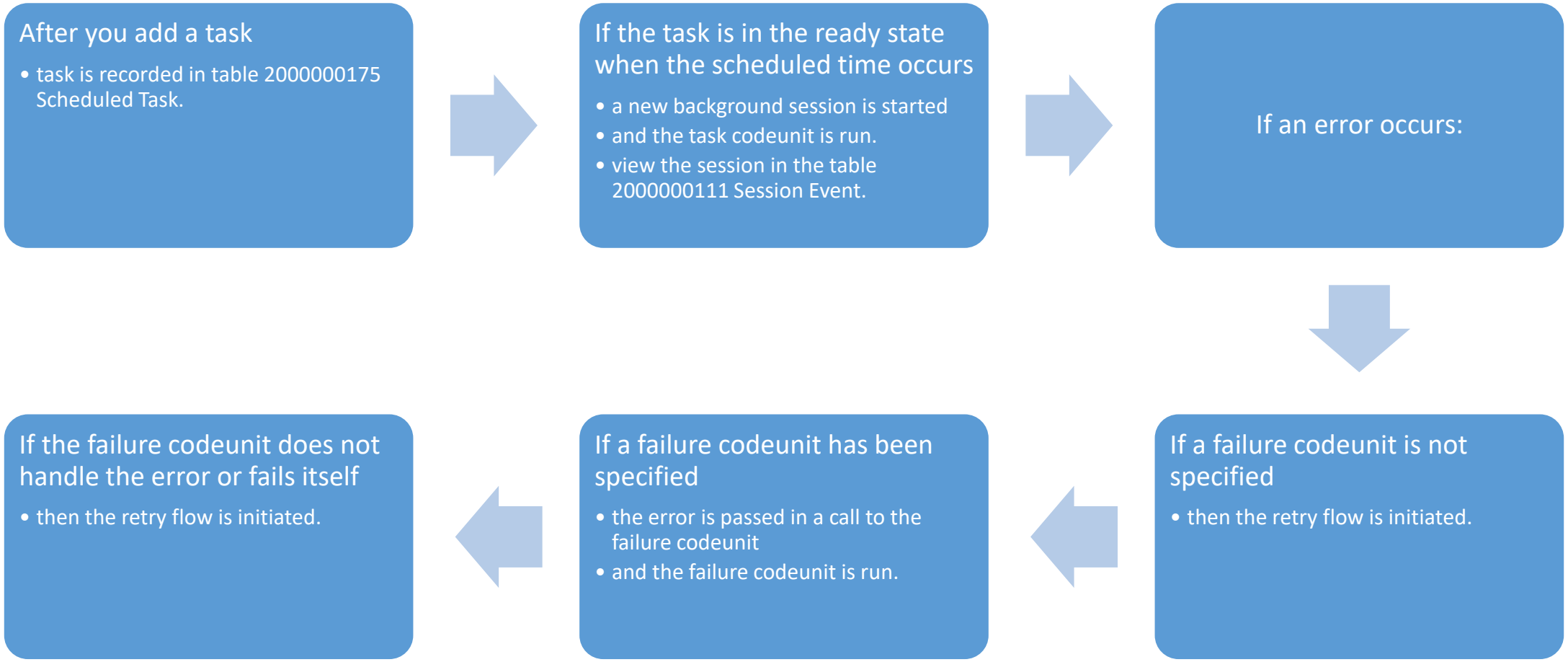
Task Scheduler

- Functions that are available for the TASKSCHEDULER data type:
  - **CREATETASK**
    - Adds a task to run a codeunit at a specified date and time.
  - **SETTASKASREADY**
    - Sets a task to the Ready state. A task cannot run until it is Ready.
  - **TASKEXISTS**
    - Checks whether a specific task exists.
  - **CANCELTASK**
    - Cancels a scheduled task.

Task Scheduler

- **To set up a task**
  - Create a **codeunit** that contains the logic that you want to schedule.
    - you can create a second codeunit (referred to as a failure codeunit)
    - that contains the logic to handle the task if an **error** occurs for any reason.
  - Once you have the codeunits
    - you can add **C/AL** code to the application
    - that calls the **CREATETASK** function
    - to **schedule** a task to run the codeunits.
  - The **CREATETASK** function
    - can also specify the **earliest date** to run the task
    - and whether the task is in the **ready** state.

Task Scheduler



Task Scheduler

A task can fail under the following conditions:

- The **company** cannot be **opened**.
- An **SQL** connection or transient **error** occurred with the database.
- The Dynamics NAV Server instance restarted **while** the task was run.

You can view these errors in the event log

- of the computer that is running the Dynamics NAV Server instance.

Task Scheduler

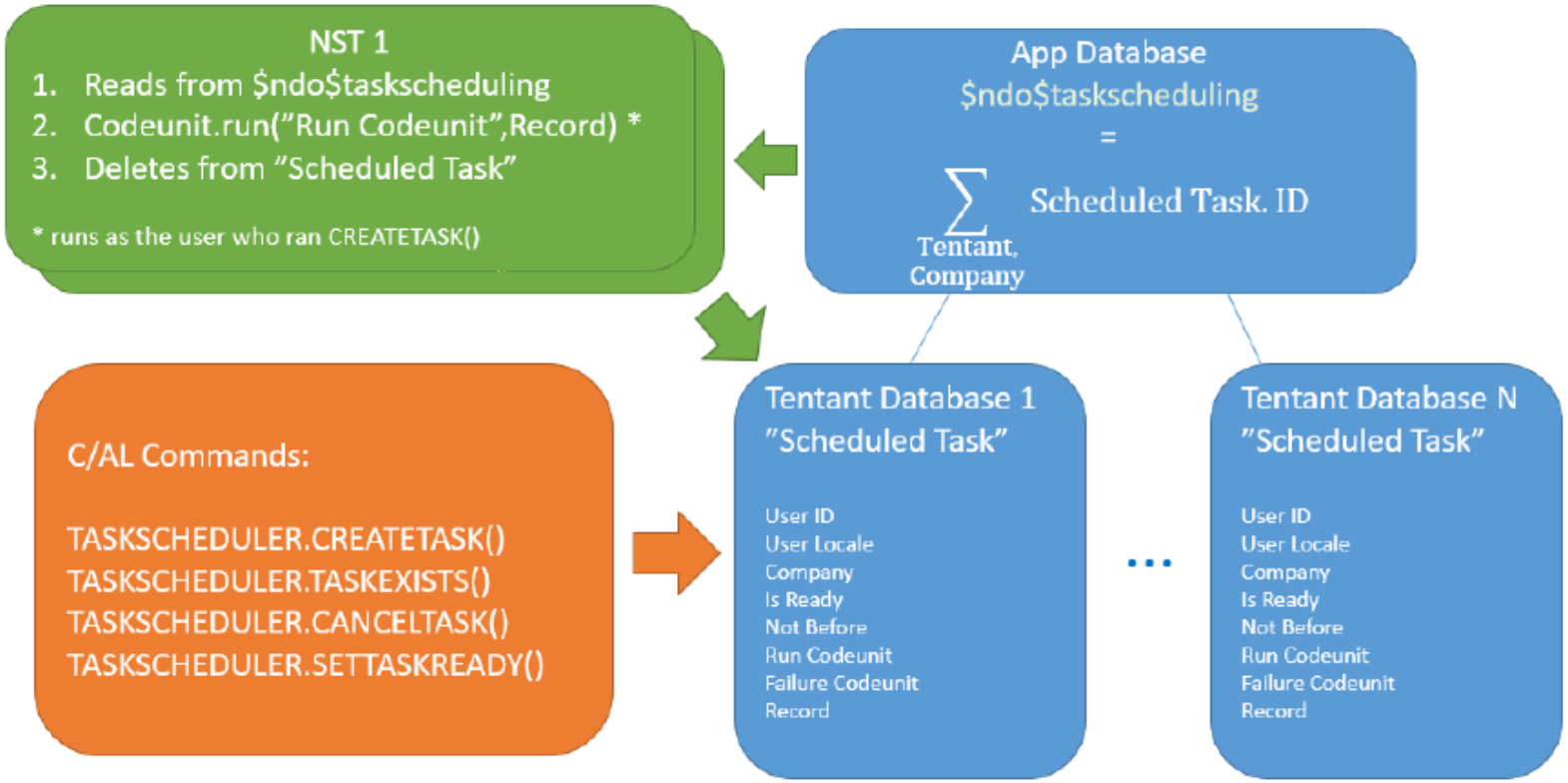
- **When an error occurs:**
  - unless the task is interrupted by the failure codeunit
  - the Dynamics NAV Server instance will rerun the task
  - according to the following retry flow:
    1. **Two** minutes after the first failure.
    2. **Four** minutes after the second failure.
    3. **Fifteen** minutes after the third failure
    4. and any subsequent failures up to a **maximum** of **10** times,
    5. after which the task is **cancelled**.

Task Scheduler

- The task runs in a background session, which means that there is **no user interface**.
  - the behavior is similar to that of the **STARTSESSION** function
  - where any **dialog** boxes that would normally appear are **suppressed**.
- The session runs by using the same **user/credentials** that are used when **calling** C/AL code.
  - The user must have appropriate **permissions** to the codeunit and any other objects that are associated with the operation of the codeunit.

Task Scheduler

New System Tables and -Functions



Task Scheduler

DynamicsNAV100 - (Running)

- General
- Database
- Client Services
- SOAP Services
- OData Services
- NAS Services
- Management Services
- Azure Key Vault Encryption Provider
- Azure Active Directory (Azure AD)
- Task Scheduler

```
<!--
  Configures the server instance to run the Task Scheduling Engine.
-->
<add key="EnableTaskScheduler" value="true"/>

<!--
  Maximum number of scheduled tasks concurrently running.
-->
<add key="TaskSchedulerMaximumConcurrentRunningTasks" value="10"/>
```

Enable Task Scheduler: ☑

Maximum concurrent running tasks: 10

Reports

Maximum concurrent running tasks
The maximum number of tasks running concurrent on the server

Edit

Task Scheduler

Table 2000000175 Scheduled Task - Table Designer

E...	Field No.	Field Name	Data Type	Length	Description	Field Class
✓	1	ID	GUID			Normal
✓	2	User ID	GUID			Normal
✓	3	User Name	Text	50		Normal
✓	4	User Language ID	Integer			Normal
✓	5	User Format ID	Integer			Normal
✓	6	User Time Zone	Text	32		Normal
✓	10	Company	Text	30		Normal
✓	11	Is Ready	Boolean			Normal
✓	12	Not Before	DateTime			Normal
✓	20	Run Codeunit	Integer			Normal
✓	21	Failure Codeunit	Integer			Normal
✓	22	Record	RecordID			Normal

Help

Task Scheduler

- Example:
  - CU 453 Enqueue

```
JobQueueEntry."System Task ID"
  TASKSCHEDULER.CREATETASK(
    CODEUNIT::"Job Queue Dispat
    CODEUNIT::"Job Queue Error H
    TRUE,
    COMPANYNAME,
    JobQueueEntry."Earliest Start
    JobQueueEntry.RECORDID);
```

Codeunit 453 Job Queue - Enqueue - C/AL Editor

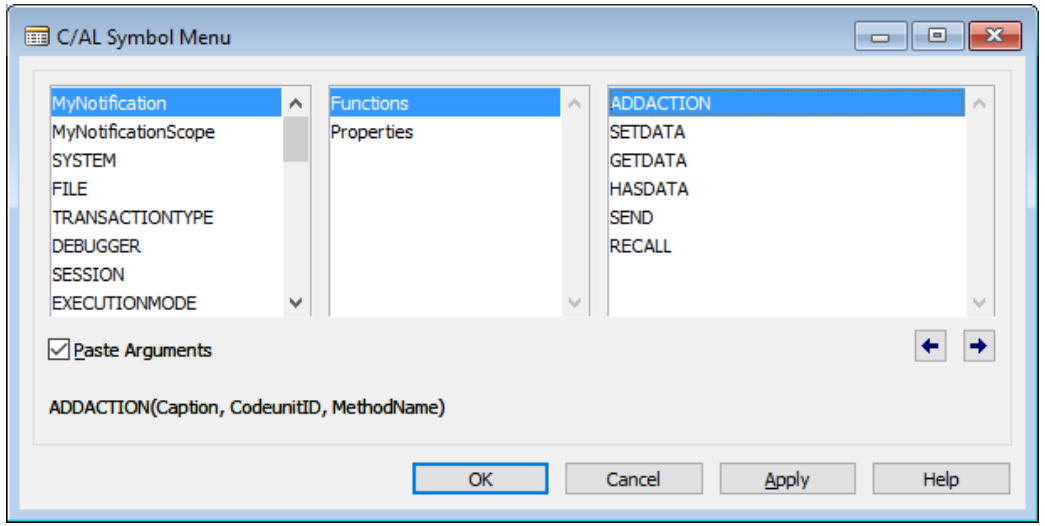
```
1  Documentation()
2
3  OnRun(VAR Rec : Record "Job Queue Entry")
4  EnqueueJobQueueEntry(Rec);
5
6  EnqueueJobQueueEntry(VAR JobQueueEntry : Record "Job Queue Entry")
7  RemoveFailedJobs(JobQueueEntry);
8  JobQueueEntry.Status := JobQueueEntry.Status::"On Hold";
9  JobQueueEntry."User Language ID" := GLOBALLANGUAGE;
10 IF NOT JobQueueEntry.INSERT(TRUE) THEN
11   JobQueueEntry.MODIFY(TRUE);
12
13 OnBeforeJobQueueScheduleTask(DoNotScheduleTask);
14
15 IF DoNotScheduleTask THEN
16   EXIT;
17
18 JobQueueEntry."System Task ID" :=
19   TASKSCHEDULER.CREATETASK(
20     CODEUNIT::"Job Queue Dispatcher",
21     CODEUNIT::"Job Queue Error Handler",
22     TRUE,
23     COMPANYNAME,
24     JobQueueEntry."Earliest Start Date/Time",
25     JobQueueEntry.RECORDID);
26 JobQueueEntry.Status := JobQueueEntry.Status::Ready;
27 JobQueueEntry.MODIFY;
28 COMMIT;
29
```

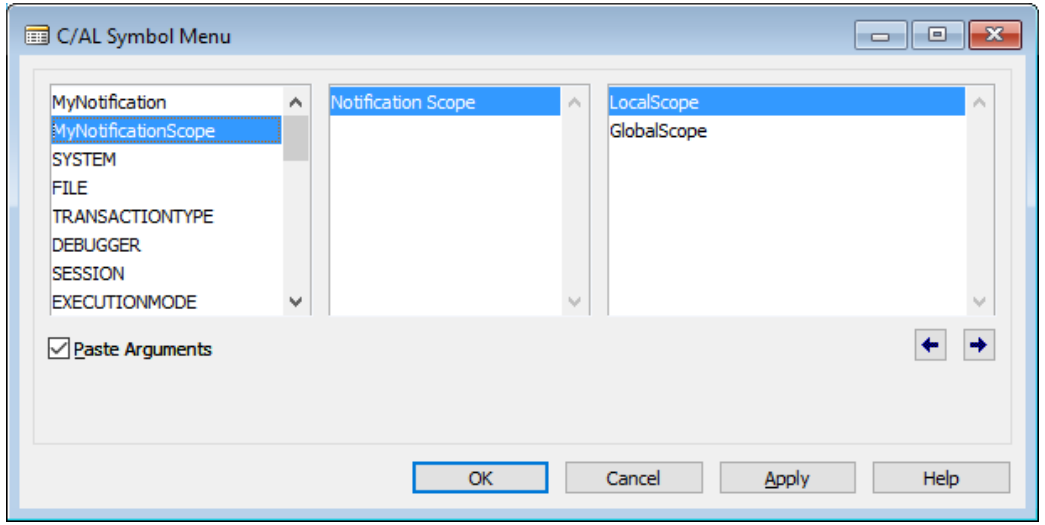
100 %

Notifications

Notifications

- The new Notification C/AL type can be used to
  - **notify** and **guide** users
  - with a subtle **message**
  - is displayed at the **top** of a page.
- Developers have
  - full **control** over the **context** in which notifications are shown,
  - can **withdraw** notifications,
  - and may **include** one or more custom **actions**.





Notifications

- Notifications provide
  - a **programmatic** way
  - to send **non-intrusive** information
  - to the user **interface** (UI)
  - in the Dynamics NAV **Web** client.

- Notifications **differ** from messages
  - initiated by the **MESSAGE** function.

Notifications

- **Messages** are **modal**
  - which means users are typically required to **address** the **message** and take some form of corrective **action** before they continue working.
- **Notifications** are **non-modal**.
  - Their purpose is to give users information about a current **situation**,
  - but do **not** require any **immediate action** or block users from continuing with their current task.
  - For example,
    - you could have a notification that a customer's credit limit is exceeded.

Notifications

View - Availability check - 1000 · Bic...

HOME | CRONUS Int...

Create Purchase Invoice | Item | View | Refresh | Clear Filter | Go to

New | Manage | Page

1000 · Bicycle

Limit totals to:

+Add Filter

The available inventory for item 1000 is lower than the entered quantity

Available Inventory: 0

Inventory Shortage: -1

Details

No.: 1000

Description: Bicycle

Earliest Availability Date:

Substitutes Exist: No

Unit of Measure Code: PCS

Close

ew - Sales Order - 1001 · Spotsmeyer's Furnishings

CRONUS International Ltd.

Statistics | Shipments | Assembly Orders | Invoices | Archive Document

Order | Documents | Order Confirmation | Posting | Request Approval | Show Attached | Page

ower than the entered quantity Details...

1/25/2018 | 1/25/2018 | 1/31/2018 | Open

ter Clear Filter

on	Location Code	Quantity	Qty. to Assemble to Order
	YELLOW	1	

Posted Sales Return Receipts:

Posted Sales Credit Memos:

Customer Details

Actions

Notifications

- Notifications in the **UI**
  - In the UI, notifications appear in the **Notification bar**
    - (similar to validation errors)
    - at the **top** of the page on which a user is currently working.
  - The user can then choose to **dismiss** the notification, which clears it.
    - Or if **actions** are defined on notification
    - the user can choose one of the **actions**.

Notifications

- There can be **multiple** notifications.
  - The notifications appear **chronological** order from top to bottom.
- Notifications **remain**
  - for duration of the page **instance**
  - or until the user **dismisses** them or takes action on them.
- Notifications that are defined on **sub-pages**,
  - for example in parts and FactBoxes,
  - appear in the **same** Notification **bar**.
- **Validation** errors on the page will be shown **first**.

Notifications

In the development environment:

- Use the
  - **Notification** and **NotificationScope**
  - data types and functions
- Add **code** to send notifications to users.

Notifications

MESSAGE	• Specifies the content of the notification that appears in the UI.
SCOPE	• Specifies the scope in which the notification appears.
SEND	• Sends the notification to be displayed by the client.
ADDACTION	• Adds an action on the notification.
SETDATA	• Sets a data property value for the notification
GETDATA	• Gets a data property value from the notification.
RECALL	• Recalls a sent notification.

Notifications

- Creating and sending a notification
  - You create a notification by using
    - the **MESSAGE** and **SEND** functions.
  - The **MESSAGE** function
    - defines the **message** part of the notification.
  - When the **SEND** function is called
    - the notification is **sent** to the client
    - and content of the message is **displayed**.

Notifications

```
MyNotification.MESSAGE := 'This is a notification';
MyNotification.SEND;
```

- The **SEND** function call should be the **last** statement in the notification code,
  - **after** any **ADDACTION** or **SETDATA** function calls.

Notifications

The scope is

- the realm in which a notification is broadcast in the client.

There are two different scopes:

- **LocalScope** and **GlobalScope**.

Notifications

A **LocalScope** notification

- appears in context of the user's current **task**,
- that is, on the **page** the user is currently working on.
- LocalScope is the **default**.

A **GlobalScope** notification

- is not directly related to the current task.
- Note: GlobalScope is currently **not supported**, so do not use it.
- This will be implemented in a **future** release.

Notifications

- The following code creates a notification in the LocalScope:

```
MyNotification.MESSAGE := 'This is a notification';
MyNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
MyNotification.SEND;
```

Notifications

- You add actions on notifications by using the **ADDACTION:**
  - Provides a way for you to create **interactive** notifications.
  - By default, users have the option to **dismiss** the notifications.

  - However, there might be cases where you want to provide users with **different actions** to address the notification:
    - like opening an associated page for modifying data.

  - A notification **action**:
    - calls a function in a specified **codeunit**,
    - passing the notification **object** in the call.
    - function includes the **business logic** for handling the action.

Notifications

```
MyNotification.MESSAGE := 'This is a notification';
MyNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
MyNotification.ADDACTION('Action 1',CODEUNIT::"Action Handler",'RunAction1');
MyNotification.ADDACTION('Action 2',CODEUNIT::"Action Handler",'RunAction2');
MyNotification.SEND;
```

Notifications

The basic steps for adding an action are as follows:

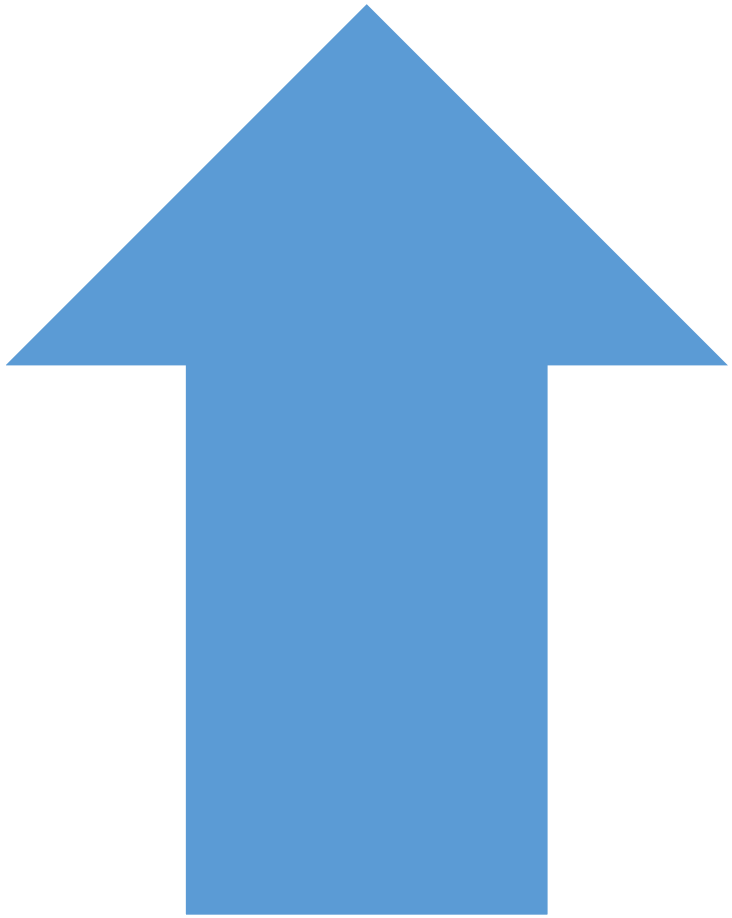
Create a **global** function in a new or existing **codeunit**.

The function must have a **Notification** data type **parameter** for receiving the notification object.

Add C/AL code to the function for **handling** the action.

Specify the codeunit and function in the **ADDACTION** function call.

Notifications



You can have **more** than **one** action on a notification.

- A **LocalScope** notification
  - can have up to **3** actions.
- A **GlobalScope** notification
  - can have up to **2** actions.

Notifications

You use the SETDATA and GETDATA functions

- to add **data** to a notification,
- which is typically needed when **actions** are invoked.

The SETDATA function

- **sets**, or **adds**, **data** to the notification.
- the data is defined as text in a **key-value** pair.

With the GETDATA function

- you can then **retrieve** the data again.

Notifications

The following code sets data for a notification:

```
MyNotification.MESSAGE := 'This is a notification';
MyNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
MyNotification.SETDATA('Created',FORMAT(CURRENTDATETIME,0,9));
MyNotification.SETDATA('ID',FORMAT(CREATEGUID,0,9));
MyNotification.ADDACTION('Action 1',CODEUNIT::"Action Handler",'RunAction1');
MyNotification.ADDACTION('Action 2',CODEUNIT::"Action Handler",'RunAction2');
MyNotification.SEND;
```

The following code gets the data for a notification:

```
DataValue := MyNotification.GETDATA('Created');
DataValue := MyNotification.GETDATA('ID');
```

Notifications

- DEMO:

Notifications

Notifications

Codeunit 1430 Role Center Notification Mgt. - C/AL Globals

Variables | Text Constants | Functions

Name
CreateAndSendEvaluationNotification
CreateAndSendTrialNotification
CreateAndSendWarningNotification
CreateAndSendSuspendedNotification
GetEvaluationNotificationId
GetTrialNotificationId
GetWarningNotificationId
GetSuspendedNotificationId
AreNotificationsSupported
IsEvaluationNotificationEnabled
IsTrialNotificationEnabled
IsWarningNotificationEnabled
IsSuspendedNotificationEnabled
ShowNotifications
ShowEvaluationNotification
ShowTrialNotification
ShowWarningNotification
ShowSuspendedNotification
EvaluationNotificationAction
TrialNotificationAction
WarningNotificationAction
SuspendedNotificationAction
IsEvaluationMode
IsTrialMode
IsWarningMode
IsSuspendedMode
GetLicenseState
GetLicenseRemainingDays
GetTrialTotalDays
StartChangeCompanyTour
BuySubscription
GetBuySubscriptionUrl
ClickEvaluationNotification
DisableEvaluationNotification
IsEvaluationNotificationClicked
ChangeCompanyTourNotAvailableMessage
StartChangeCompanyTourQuestion

Codeunit 1430 Role Center Notification Mgt. - C/AL Editor

```
5   LOCAL CreateAndSendEvaluationNotification()
6   TrialTotalDays := GetTrialTotalDays;
7   EvaluationNotification.ID := GetEvaluationNotificationId;
8   EvaluationNotification.MESSAGE := STRSUBSTNO(EvaluationNotificationMsg,TrialTotalDays);
9   EvaluationNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
10  EvaluationNotification.ADDACTION(
11    EvaluationNotificationLinkTxt,CODEUNIT::"Role Center Notification Mgt.",'EvaluationNotificationAction');
12  EvaluationNotification.SEND;
13
14  LOCAL CreateAndSendTrialNotification()
15  RemainingDays := GetLicenseRemainingDays;
16  TrialNotification.ID := GetTrialNotificationId;
17  TrialNotification.MESSAGE := STRSUBSTNO(TrialNotificationMsg,RemainingDays);
18  TrialNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
19  TrialNotification.ADDACTION(
20    TrialNotificationLinkTxt,CODEUNIT::"Role Center Notification Mgt.",'TrialNotificationAction');
21  TrialNotification.SEND;
22
23  LOCAL CreateAndSendWarningNotification()
24  RemainingDays := GetLicenseRemainingDays;
25  WarningNotification.ID := GetWarningNotificationId;
26  WarningNotification.MESSAGE := STRSUBSTNO(WarningNotificationMsg,RemainingDays);
27  WarningNotification.SCOPE := NOTIFICATIONSCOPE::LocalScope;
28  WarningNotification.ADDACTION(
29    WarningNotificationLinkTxt,CODEUNIT::"Role Center Notification Mgt.",'WarningNotificationAction');
30  WarningNotification.SEND;
31
```

100 %

Notifications

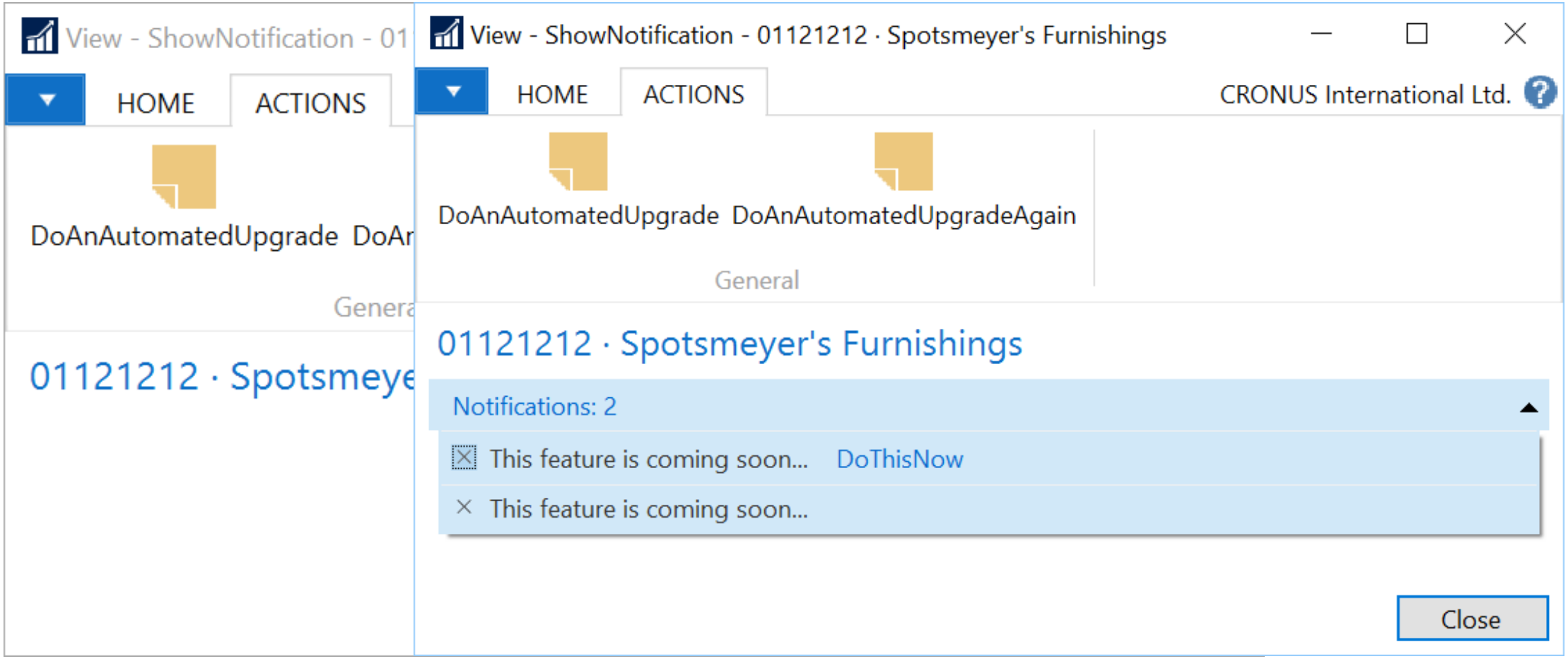
Microsoft Dynamics NAV

Page 1871 Credit Limit Details - C/AL Editor

```
136 PopulateDataOnNotification(VAR CreditLimitNotification : Notification)
137 CreditLimitNotification.SETDATA(FIELDNAME("No."),FORMAT("No."));
138 CreditLimitNotification.SETDATA('OrderAmountTotalLCY',FORMAT(OrderAmountTotalLCY));
139 CreditLimitNotification.SETDATA('ShippedRetRcdNotIndLCY',FORMAT(ShippedRetRcdNotIndLCY));
140 CreditLimitNotification.SETDATA('OrderAmountThisOrderLCY',FORMAT(OrderAmountThisOrderLCY));
141 CreditLimitNotification.SETDATA('CustCreditAmountLCY',FORMAT(CustCreditAmountLCY));
142
143 InitializeFromNotificationVar(CreditLimitNotification : Notification)
144 GET(CreditLimitNotification.GETDATA(Customer.FIELDNAME("No.")));
145 SETRANGE("No.","No.");
146
147 IF GETFILTER("Date Filter") = '' THEN
148   SETFILTER("Date Filter",'..%1',WORKDATE);
149 CALCFIELDS("Balance (LCY)","Shipped Not Invoiced (LCY)","Serv Shipped Not Invoiced(LCY)");
150
151 EVALUATE(OrderAmountTotalLCY,CreditLimitNotification.GETDATA('OrderAmountTotalLCY'));
152 EVALUATE(ShippedRetRcdNotIndLCY,CreditLimitNotification.GETDATA('ShippedRetRcdNotIndLCY'));
153 EVALUATE(OrderAmountThisOrderLCY,CreditLimitNotification.GETDATA('OrderAmountThisOrderLCY'));
154 EVALUATE(CustCreditAmountLCY,CreditLimitNotification.GETDATA('CustCreditAmountLCY'));
155
```

100 %

Notifications



Time Series API

Time Series API

- Dynamics NAV 2017 enters the world of **machine learning** and the first step of this journey is to bring the **Time Series API** to NAV Developers.

- Let me start with a short explanation of what **machine learning** (predictive analytics) means in general.

- The current way of building software is:
  - A smart guy creates an **algorithm**.
  - A **user inputs** some value,
  - The system applies the algorithm and returns some **results**.

Time Series API

- To create a good algorithm
  - you must be a domain **expert**
  - and have deep **knowledge** of a specific industry.

- The worst part here is that
  - someone must **update** the algorithm programmatically
  - or by changing the **setup**
  - on a regular basis to adapt to **changeable** business **processes**.

Time Series API

- Machine learning helps you do things in a **different** way:
  - It takes **historical data**, automatically **analyzes** it in different ways, and then tries to **find** the **algorithm** by itself.
    - (Model or Experiment in Azure ML)
  - The **user inputs** some value, **system applies** the **model** and returns **results**.
  - The system uses the **new** user input to **adjust** the **model**.

Time Series API

- Actually, this is **not** completely **true**
  - because if the system tries to analyze the incoming data in different ways, it will be very **expensive**.
  - Each attempt requires significant computation **efforts**.
  - That is why this approach is **not** used.

- Instead, the domain **expert** helps the system
  - by **advising** between the available algorithms,
  - and **system** does the **rest** of the work.

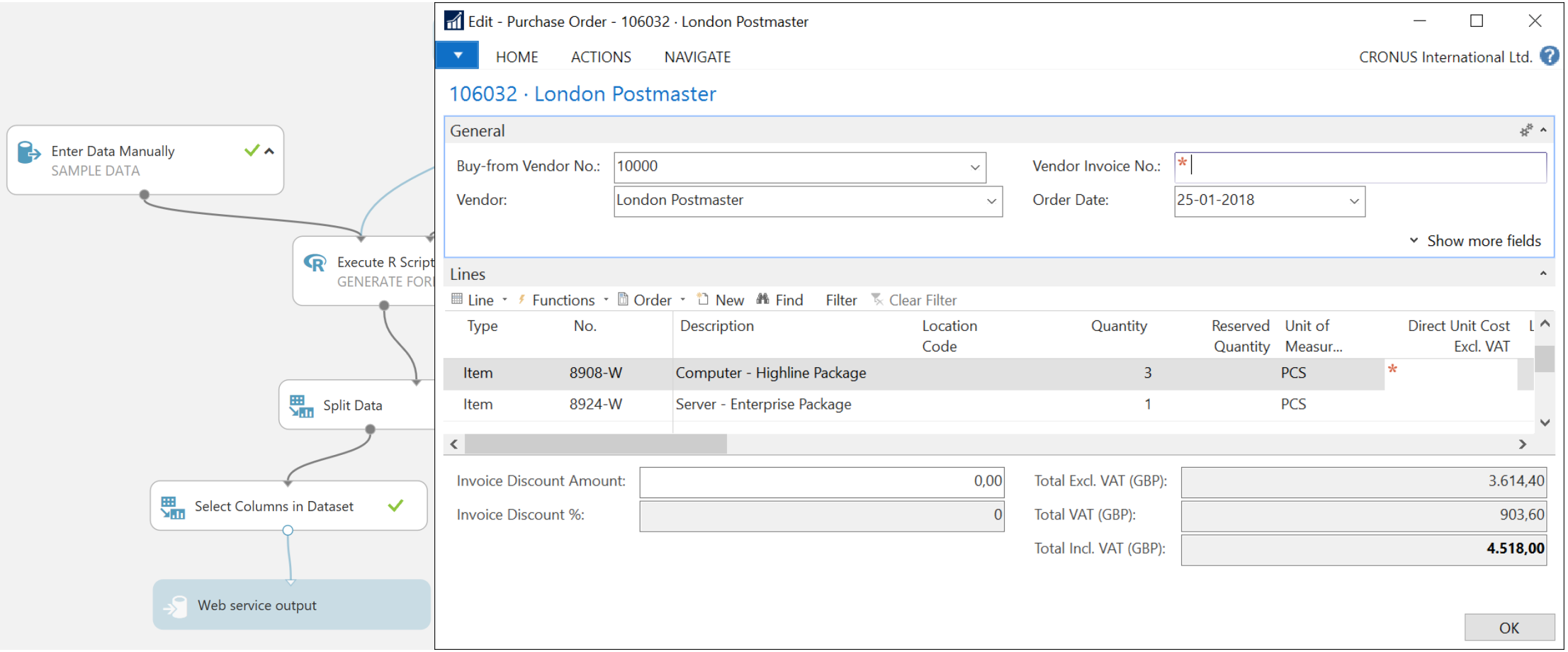
Time Series API

- One of the most popular topics for machine learning is **forecasting** based on historical data.
  - Many algorithms that can do this.
- We chose **five** of these algorithms,
  - wrapped them in **one** Azure ML **experiment**,
  - added **logic** that **compares** different **results**
  - and **returns** the **best** one
  - accompanied by an **indicator** of the prediction's **quality**.
- There is a **generic API** on top of this
  - that allows Dynamics NAV developers create their own functionality
  - that helps customers find the business-critical information
  - that may be hidden in their database.

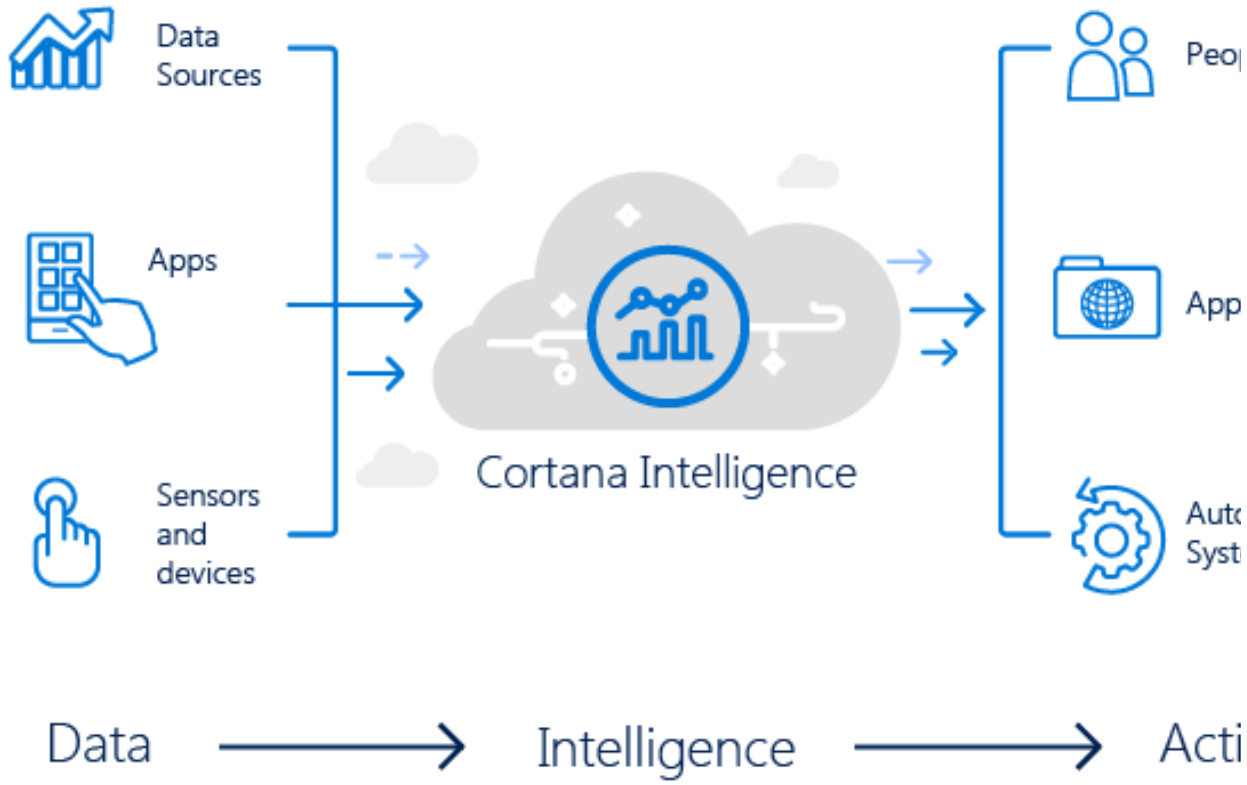
Time Series API

- This is the best **API** for the following reasons:
  - The current implementation does **not** require you to **re-train** the model
    - as this is done when the request is **submitted**.
  - More than **300** tables contain both Date and Decimal fields in Dynamics NAV.
    - This is the most typical table structure in business applications (28%)
    - They are good candidates for analysis with a Time Series API.

Time Series API



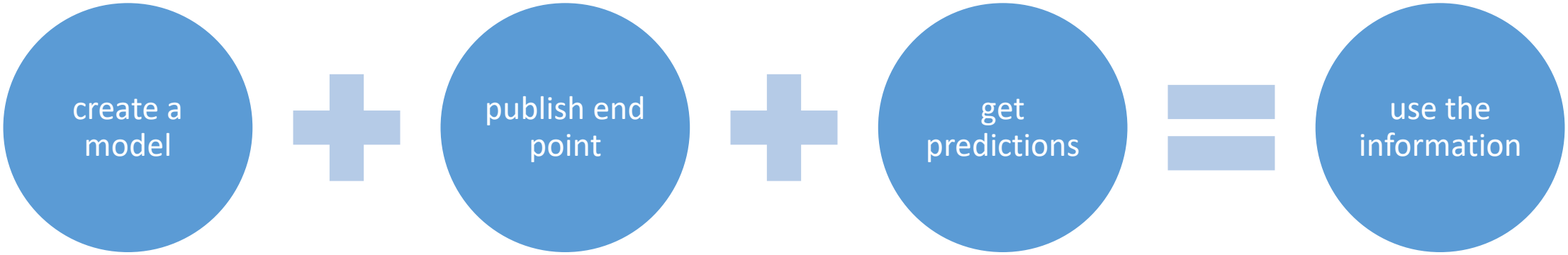
Time Series API



CORTANA INTELLIGENCE SUITE

Capabilities		Products
Preconfigured solutions	• Business scenarios	• Forecasting, churn, etc.
Intelligence	• Integration with Cortana • Bot services • Cognitive services	• Cortana • Bot Framework • Cognitive Services
Dashboards and visualizations	• Dashboards and visualizations	• Power BI
Machine learning and advanced analytics	• Machine learning • Hadoop • Distributed analytics • Complex event processing	• Machine Learning • HDInsight (Data Lake service) • Data Lake analytics • Stream Analytics
Big data stores	• Big Data repository • Elastic data warehouse	• Data Lake store, Blobs • SQL Data Warehouse
Information management	• Data orchestration • Data catalog • Event ingestion	• Data Factory • Data catalog • Event Hubs

Time Series API



Time Series API

- **Create a model**
  - time-series predictions model:
    - https://gallery.cortanaintelligence.com/Experiment/Dynamics-Ax7-demand-forecasting-1
  - open in Azure Machine Learning studio
  - copy experiment to your workspace
  - validate the experiment

Time Series API

- **Publish an end point**
  - deploy web service
  - API key
  - request URI

Time Series API

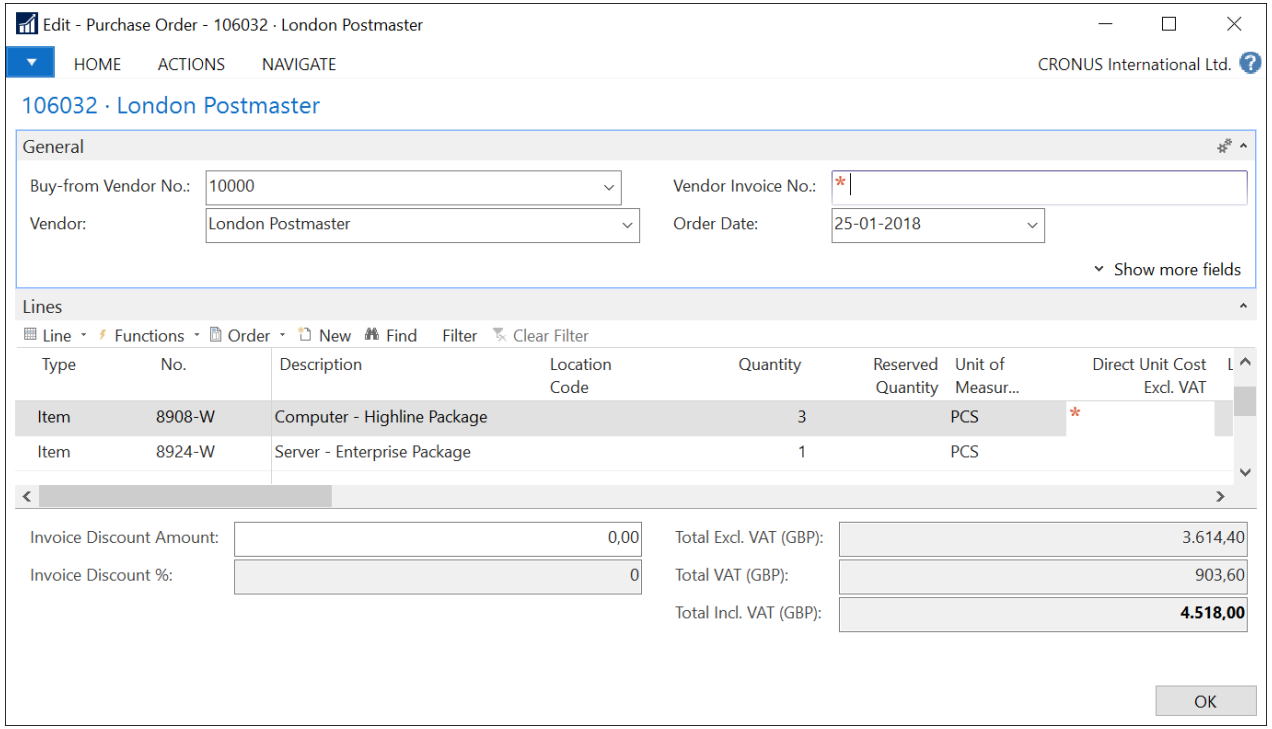
- **Get predictions**
  - call the Time Series **API** from the development environment
  - to get predictions on different data
  - **check** the **quality** of the predictions programmatically

Time Series API

- **Time Series Library:**
  - **Initialize**
    - set up the connection
  - **PrepareData**
    - transforms any table data into a dataset
  - **Forecast**
    - calls the Azure ML
  - **GetForecast**
    - returns a dataset of forecasted values

Time Series API

- **Use the information**
  - create a purchase order



Time Series API

- DEMO

Resources

Resources

Microsoft Dynamics NAV Get Ready Page on Partner Source

- https://aka.ms/NavGetReady

Microsoft Dynamics Learning Portal

- https://aka.ms/DynamicsLearningPortal

Developer and IT-Pro Help for Microsoft Dynamics NAV

- https://msdn.microsoft.com/en-us/dynamics-nav/index

How Do I Video's:

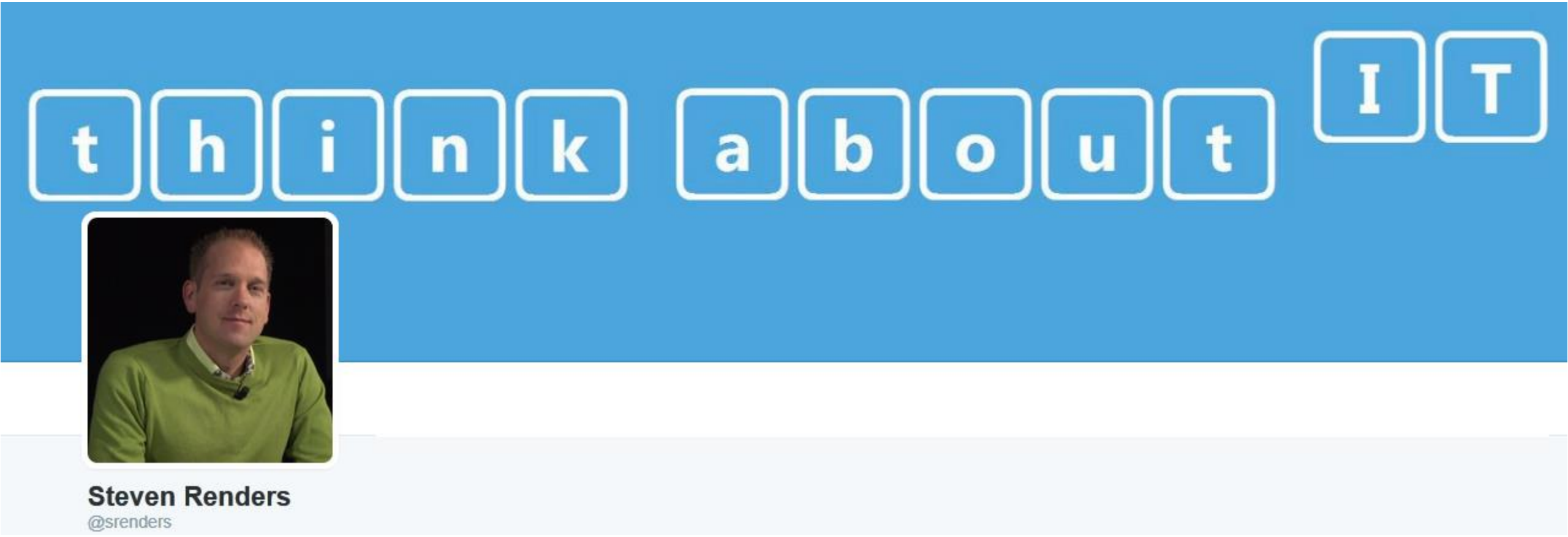
- https://www.youtube.com/playlist?list=PL5B63EF419A3B59C8

Review

- Assisted Setup
- Page Wizard
- Working With Media on Records
- Task Scheduler
- Notifications
- Time Series API
- Resources / Help

Summary

- Steven Renders
  - Steven.Renders@thinkaboutit.be




Happy
Holidays!
DW4.ME