

# Zetadocs SDK Guide





# Table of Contents

<b>1. The Zetadocs Software Development Kit</b>	<b>3</b>
1.1 Point Solutions .....	5
<b>2. Zetadocs Delivery Plus</b>	<b>7</b>
2.1 Configured Send .....	9
2.2 Custom Send .....	11
2.3 Custom Addressing .....	15
2.4 Custom Delivery Options .....	19
2.5 Delivery Automation .....	27
<b>3. Zetadocs Capture Plus</b>	<b>30</b>
3.1 Understanding the Capture Plus SDK Framework .....	30
3.2 The Zetadocs Server .....	32
3.3 Zetadocs Server Setup .....	34
3.3.1 Adding a New Document Queue .....	36
3.3.2 Enabling and Disabling Abbyy .....	37
3.3.3 Batch Scanning Settings .....	37
3.3.4 Barcode Settings .....	38
3.4 Custom Document Queues .....	39
3.4.1 Adding Document Queues to the MenuSuite .....	40
3.4.2 Archive Search and Retrieval .....	41
3.4.3 Implementing Custom Document Queue Actions .....	41
3.4.4 Testing Document Archiving .....	45
3.5 Automation Using Barcodes .....	45
3.5.1 Installing the Barcode Font .....	47
3.5.2 Modify an NAV Report to Add a Barcode .....	47
3.5.3 Adding Code to the GetLinkDisplayString Function .....	48
3.5.4 Write Code in the GetAutoLink Function .....	50
3.5.5 Testing Document Auto Linking .....	52
3.6 Automation Using Extracted Text .....	52
3.6.1 Approvals in NAV using Extracted Text .....	53
3.6.2 Creating the GetMatchingPurchaseOrder function .....	55
3.6.3 Adjusting the GetAutoLink Function .....	56
3.6.4 Adjusting the PostArchive Function .....	57
3.7 Further Capture Customizations .....	58
3.8 Capture Reference .....	59
3.9 Capture Automation .....	64
3.9.1 Enter the archiving credentials in SharePoint .....	64
3.9.2 Call Zetadocs Capture Interface .....	65
3.9.3 Test your system .....	67

## 4. Further Resources

68

<b>4.1 How to add the Zetadocs Document FactBox to records outside Sales and Purchasing .....</b>	<b>68</b>
<b>4.1.1 Introduction .....</b>	<b>68</b>
<b>4.1.2 Add Zetadocs Record Mappings in Service Quotes and Service Orders .....</b>	<b>69</b>
4.1.2.1 Add the FactBox with Metadata to pages e. g. Service Orders & Service Quotes .....	70
4.1.2.2 Add the FactBox to items which does not have a page e. g. Service Item Lines .....	71
4.1.2.3 See additional documents in the FactBox .....	73
4.1.2.3.1 Relate Service Item Lines to Service Orders .....	73
4.1.2.3.2 Relate Service Quotes to Service Orders .....	75
<b>4.1.3 Add the Zetadocs FactBox to the General Ledger Entries page .....</b>	<b>77</b>
<b>4.1.4 See additional documents related to Jobs in Sales Invoices and Posted Sales Invoices .....</b>	<b>79</b>
<b>4.2 Troubleshooting .....</b>	<b>81</b>



# 1. The Zetadocs Software Development Kit

The Zetadocs SDK is made up of two parts, Zetadocs Capture Plus and Zetadocs Delivery Plus. It was created to allow Microsoft Dynamics developers to further customize the Zetadocs features to meet bespoke requirements. With the SDK it is easy to leverage the core features of Zetadocs, Delivery and Capture, to provide complex, fully integrated, high value business processes. On ordering your Zetadocs solution you may have purchased one of our [point solutions](#), these are regularly applied customizations which we use to demonstrate the kinds of solution you can create. Whilst we have provided details explaining how these solutions can be implemented we will also be covering a number of other options which allow you to customize Zetadocs still further. Please note this guide is designed to provide you with the steps you need to use the SDK to customize an existing Delivery or Capture Essentials system. For brevity, this guide will refer to Microsoft Dynamics 365 Business Central on-premises as BC and Microsoft Dynamics NAV versions as NAV plus their respective version e.g. NAV 2018.

**NOTE:** To get the most from the Zetadocs SDK it is recommended that you take a moment to understand the goals of the SDK and the logic behind the framework. This section aims to give a short overview of the SDK framework which will then simplify your understanding of the Capture and Delivery Plus examples in the subsequent sections and help you in designing good solutions with Zetadocs.

## Goals of the SDK

It is common in the NAV and BC community that customizations are implemented as and where the developer sees fit. Most NAV and BC objects are open to developers to modify. The positive aspect of this is the speed of implementation. However there is a negative aspect and that is the cost of maintenance and upgrade. Equisys identified the challenge this presents for developers and has designed this SDK with this in mind. Our goal is to make upgrades and patches of the Zetadocs product much simpler and faster by removing the need to merge customizations into updated object versions. Our SDK goals are:

1. Structured - Provide designated entry points for NAV and BC developer code
2. Flexible - Integrate Zetadocs features with any records of NAV or BC\*
3. Intuitive - Use a framework to provide a consistent style of customization implementation

*\*This refers to the use of the typeless RecordID and RecordRef types over the strongly typed Record type. There is no guarantee of the suitability of any NAV or BC record for use with Zetadocs.*

## Understanding the Framework

There are three elements to the Zetadocs SDK. They are:

1. Interfaces
  - This is the name given to existing NAV or BC forms, pages, codeunits or reports that are modified to call the Zetadocs Core codeunits to provide Zetadocs functionality. Our aim is to make these modifications as minimal as possible.
2. Zetadocs Core Codeunits

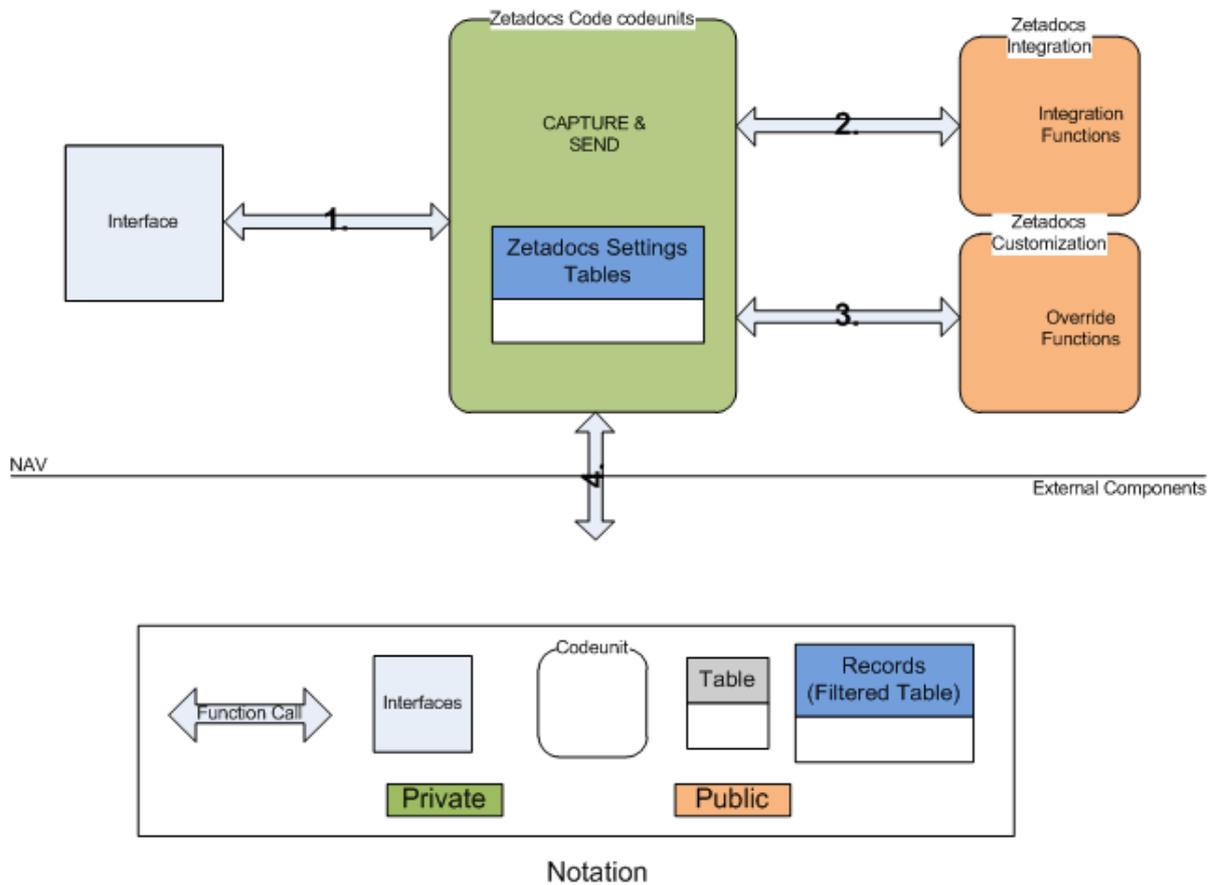
- These are our codeunits which developers do not have modify permissions for. They perform the bulk of Zetadocs processing and call the Integration and Customization codeunits to run developer code.
3. Integration and Customization Codeunits
    - These are our codeunits which developers with the Zetadocs SDK granule do have modify permissions for. They provide simple call out functions which the developer implements his custom code in.

Developers should look to understand the integration and customization entry points before designing their solution.

## Process Flow

The processing flow follows the same pattern in both Plus feature areas.

1. The interfaces calls the Zetadocs Core codeunits which use the Zetadocs settings to perform the action.
2. If there is no suitable configuration available for the records being processed then the Zetadocs Code calls out to the Integration and Customization codeunits to get the information. It is here that the developer can implement bespoke settings and integrate with custom records.
3. Once all processing has been completed, the Zetadocs Core calls out to the Customization codeunits with modifiable data sets to instruct the core codeunits on how to complete the action processing. It is here that the developer can implement custom business logic, enhanced Zetadocs feature functionality and integration with other subsystems.
4. Finally, having processed the developer modifications, the Zetadocs Core codeunits call outside of NAV or BC to use the Zetadocs Client components to complete the functionality.



Zetadocs SDK Architecture

## 1.1 Point Solutions

Zetadocs point solutions are examples of common customizations which we use as examples of the kinds of solution that can be created using the SDK. We use these to enable VARs to identify and license the components they need to achieve certain effects.

### Supplier Invoice Processing

The Zetadocs Supplier Invoice Processing point solution allows your received supplier invoices to be automatically scanned and archived, as soon as an invoice has been entered onto NAV or BC, a barcoded label is produced by a small desktop printer for sticking onto the invoice. It can then be scanned and automatically linked to the relevant transaction in NAV or BC.

The following actions are required to achieve this point solution, these are explained throughout the SDK but to view an example of how this can be setup please view the technical note available [here](#).

1. Configuring the Zetadocs Server
2. Installing the barcode font
3. Creating a report that prints a barcode
4. Add a button to run the print barcode report
5. Add the AutoLink code
6. Creating a Document Queue to run the GetAutolink code
7. Add the Document Queue to the NAV or BC menu suite (in Development Environment if required)
8. Test Auto Linking

## Proof of Delivery

The Zetadocs Proof of Delivery point solution allows documents to be created with a barcoded unique identifier, that allows the document to be automatically identified, archived and indexed against the relevant record in NAV or BC upon its return. Once stored, the POD can be quickly and easily retrieved from one central place, either within NAV or BC or from the archive along with other key information such as the sales order or invoice.

By using Proof of Delivery you can automate and streamline this process significantly, firstly on printing the Shipment note from NAV or BC, Zetadocs can add a Barcode to the document. It is then signed as usual by the customer but on return it is simply scanned. Zetadocs can then recognise the barcode and automatically adds a PDF copy of the document to your archive and associates it with the original order that the Shipment note was based upon. This is done using a custom Shipment Document Queue. It gives the user processing these documents a single location from where they can view the signed Shipment Note and create and send the corresponding invoice. This provides the organisation with an archive that is populated with related records e.g. a Quote, Order, Shipment Note, Returned Signed Shipment Note, Invoice which enables them to view the whole transaction workflow from the archive. The sample code to achieve this is shipped as part of the Zetadocs-Capture Customize codeunit (9009964).

The following actions are required to achieve this point solution, these are explained throughout the SDK but to view an example of how this can be setup you should read the [Zetadocs Capture Plus](#) section which uses a Shipment Document Queue setup as an example.

1. Shipment Document Queue Configuration
2. Adding Document Queues to your Menu suite
3. Adding Document Queues to a Role Centre
4. Modify record forms for archive search and retrieval
5. Implementing your Custom Document Queue Actions
6. Test document archiving



## 2. Zetadocs Delivery Plus

The Zetadocs Addin can be split into 2 main areas, data and processing. During the Zetadocs Essentials install we saw how reports could be grouped into Document Sets which share a common rule for formatting and archiving. Further we saw how rules can be stored which alter formatting and sending information dependent on who we were sending to. It is assumed then that we have a good understanding of the information involved in sending a report using Zetadocs. In this architectural overview the process of collating the sending information, which we refer to as the “Send Results”, is explained. This includes a breakdown of the main processing areas and their use in integrating with NAV or BC as well as an insight into the customization potential and extensibility of Zetadocs.

### Understanding the SDK Framework

Before continuing it is recommended that you read the [Zetadocs SDK introduction](#) section of this document.

The three elements to the Zetadocs Delivery Plus SDK are:

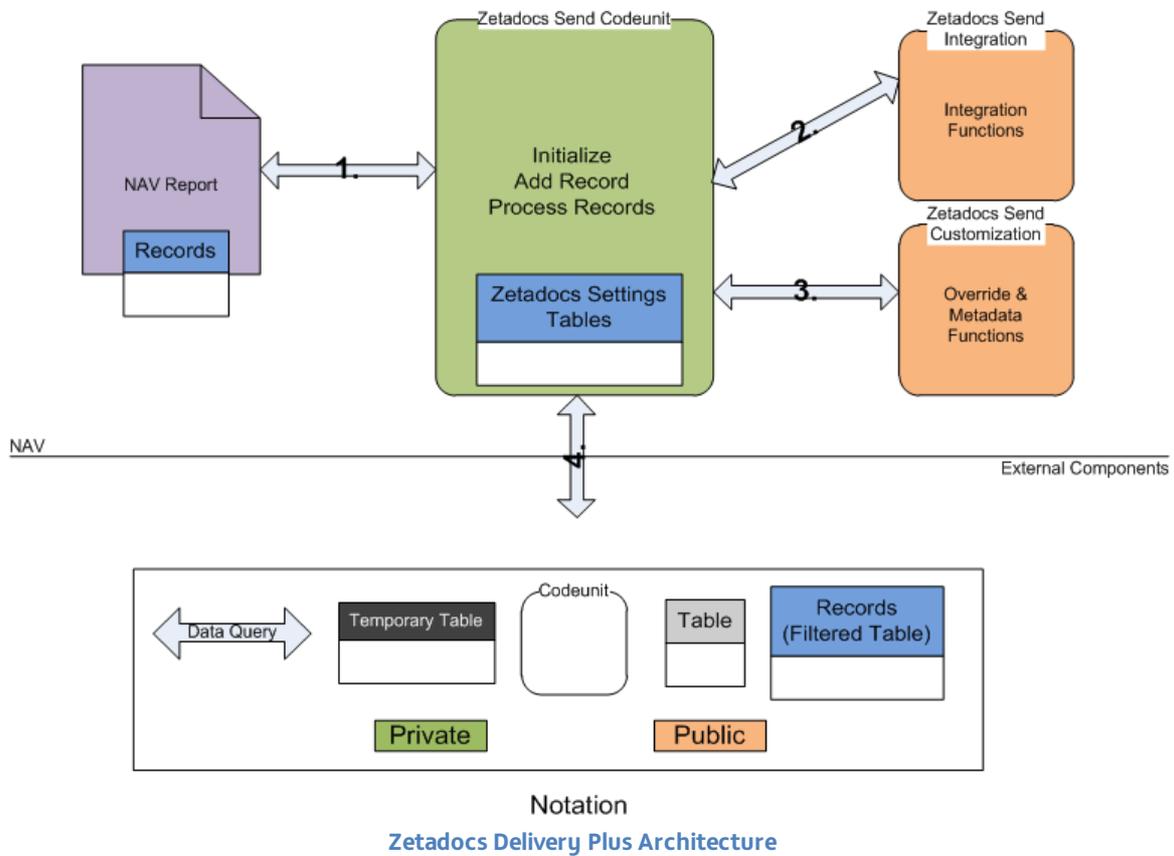
1. Interfaces
  - NAV or BC Reports
    - Reports need to be modified to call the Zetadocs Core sending codeunits.
2. Zetadocs Core Codeunits
  - Zetadocs Send codeunit
    - This codeunit provides functions that are called by the Interfaces.
3. Integration and Customization Codeunits
  - Zetadocs Send Integration codeunit
    - This codeunit provides functions called by the Zetadocs Core codeunits to integrate Zetadocs Delivery functionality into bespoke records or other areas of NAV or BC beyond Sales and Purchase Order Processing
  - Zetadocs Send Customize
    - This codeunit has functions called by the Zetadocs Core codeunits to allow the developer to implement bespoke business logic and delivery and archive settings.

### Process Flow

Now let's look at the flow of data that occurs when sending a report. The following diagram helps to visualize the process and will aid us in discussing the relevant parts of the product.

As you can see from the diagram it starts with a NAV or BC report that has been modified for Zetadocs. A report is generally based on a set of records that are used to produce some report output, a document for each record. The records contain information that relates to a contact and possibly a company, to whom the report should be sent (i.e. the recipient). Sometimes that information can be a direct table reference to an existing table that represents a contact or company for example, or it may contain some other form of references which can be used to find the contact information from another source. This could be a look up table of some sort based on some bespoke logic.

The Zetadocs Send Results Gen codeunit is the central processing controller and builds a runtime list of send results in a temporary table of type “Zetadocs Send Result” for each record in the report.



**Interfacing with the Zetadocs Send Results Gen codeunit (1.)**

A report modified for Zetadocs will call the three trigger functions on the Zetadocs Send codeunit. Firstly, **Initialize (1.)** is called to set the report that is being printed. The initialization information allows us to determine what report settings, if any, have been specified (2.). After initialization we start to process individual record specifics. Each record reference is passed to the send results codeunit by calling **AddRecord (1.)**. Each record is then processed to extract the contact information (3.). Depending on who the company is we may apply specific rules covering how the contact or the template is determined. Once all the records have been added **Process Records (1.)** is called to start finalizing the results. When all rules and settings have been applied the “Zetadocs Customize” codeunit is called to offer overriding of the result (4.). Lastly, Zetadocs Send code will run to offer runtime user result override and error resolution dialogs.

**Querying the Zetadocs Settings (2.)**

Based on the ID of the report that is being printed we can determine what template and archiving settings to use for the report and potentially how to automatically find the company and contact references from the report records. Internally the Zetadocs Send Results Gen codeunit will query the Zetadocs Report Settings using the report ID. These settings control the flow of processing from this point on.



the respective Contact reference. With this information Zetadocs was able to extract the references and query the names, email addresses and fax addresses of the contact, as well as, cross reference the company (Customer or Vendor) reference with the associated Zetadocs Customer or Vendor rules tables to apply per company formatting and addressing logic. This however is not the limit of Zetadocs functionality. This principle of field reference and related information resolution can be applied to almost any report. In this section we will see how small changes in the report settings can control and alter the information used to address, format, send and archive a report using Zetadocs.

It should be clear by now from the installation guide that by setting the “Company Type Code”, “Company No. Field No.” and “Contact No. Field No.” fields in the Zetadocs Report Settings that you can configure Zetadocs to automatically resolve your contact information and apply company specific rules if they exist. In this way it is possible to reference any fields which contain the right type of reference (i.e. customer/vendor or contact etc). There is some logic to the application of these settings that is important to know to help decide what settings to enter but it also gives the opportunity for some variations in configuration.

### Zetadocs Report Settings Field No.’s Reference Logic (Pseudo Code)

```
IF Company Type Code <> NULL THEN
  IF Company No. Field No. <> NULL THEN
    Get the Company No. from Record using Field No.
  ELSE
    CALL Integration codeunit GetCompanyForReport for VAR to get Company No.
  END
END
END
```

Send Result.VALIDATE(Company No., Company No. Value)

```
IF Contact No. Field No. <> NULL THEN
  Get the Contact No. from Record using Field No.
ELSE
  CALL Integration codeunit GetContactForReport for VAR to get Contact No.
END
```

Send Result.VALIDATE(Contact No., Contact No. Value)

From this it is important to notice the order that validation happens in as this coordinates with the Zetadocs Send Result table **OnValidate** trigger code for “Company No.” and “Contact No.”. Since the company information is validated first no contact reference will be set yet. We can check for this and default to using the company’s communication information, later when the contact reference is validated we can override the addressing information (email, fax). This provides the fallback addressing logic of Zetadocs so that if a record has a blank reference for a contact we will still end up with the company correspondence information. This means that we can deliberately control what information is used by altering the settings. It also means that this behavior can be customized by the VAR by implementing the get trigger functions in the Zetadocs-Send Integration codeunit and/or changing the OnValidate triggers in the Zetadocs Send Result table.

Company Type Code	Company No. Field No.	Contact No. Field No.	Resulting Addressing Info Used
-------------------	-----------------------	-----------------------	--------------------------------

SET	SET	SET	Contact Info unless the reference in record is blank THEN Company Info
SET	NULL	SET	Contact Info unless the reference in record is blank THEN Company Info using Company Reference implemented by VAR
SET	NULL	NULL	Contact Info using Contact Reference Implemented by VAR unless that reference is blank THEN Company Info using Company Reference implemented by VAR
NULL	NULL	SET	Contact Info
NULL	SET	SET/NULL	Invalid – You must specify a company type to enable use of company field.

To get a strong understanding of this it is recommended that you review the OnValidate triggers for “Company No.” and “Contact No.” in the Zetadocs Send Result table and familiarize yourself with the trigger functions in the Zetadocs-Send Integration codeunit. These are explained in detail in the next section.

## 2.2 Custom Send

We saw in the last section how selecting certain report settings can control the flow of the operation of the Zetadocs processing codeunits. By exploiting these settings and implementing the trigger functions in the Zetadocs-Send Integration codeunit we can highly customize our report sending. In this section we will focus on the trigger functions in the Zetadocs-Send Integration codeunit and explain how they can be used.

### GetZetadocsArchiveID

This function has been deprecated in version 5.0 of Zetadocs and remains in the code for backward compatibility. For system upgrades please read the warning below.

**WARNING:** This function should be left unchanged. It is only exposed in this codeunit to allow a VAR to add additional permissions to the codeunit for reading and writing the Zetadocs Archive ID value to and from additional tables!

## OverrideSendSettings

Apart from setting up the company and contact field number references in the Zetadocs Report Settings there are a number of other options that control the display options and the report batch information settings. During the **Initialize** function (See Understanding the SDK Framework) these settings are copied in to a temporary table called the Zetadocs Send Settings. This gives us the ability to adjust these settings at runtime. Once the report settings have been copied into the temporary send settings table described, the Zetadocs Send Results Gen codeunit calls the **OverrideSendSettings** trigger function in the Zetadocs-Send Integration codeunit. Here you can modify the settings based on some logic for that send instance.

- Report ID: **WARNING:** This is not for editing. It is here to allow the developer to determine which report is being printed in the trigger and to branch CASE or IF statements depending on its value.
- Batch Name: In the Zetadocs Report Settings you have the ability to specify Default Batch Name. This is the name that will be displayed in the Zetadocs Client for the report records printed to Zetadocs. This default value can be overridden in this function.
- Show Delivery Dialog, Show batch Delivery Dialogs, Preview in Zetadocs Client and Test Mode: All these can be dynamically overridden in this trigger at runtime.
- Company Type Code: Similar to the Report ID, this value is not for editing but is present to allow the developer to determine the target company type configured for this report without having to query the Zetadocs Report Settings Table.

A good example of when you might use this function is if you wish to have a report run uninterrupted by UI dialogs and previewing in the Zetadocs Client if it is being sent during the night. All previews could be turned off.

E.. Field No.	Field Name	Data Type	Length	Description
1	Report ID	Integer		
2	Batch Name	Text	120	
3	Show Delivery Dialog	Boolean		
4	Show Error Dialog	Boolean		
10	Company Type Code	Code	20	
20	Client Preview	Boolean		
30	Test Mode	Boolean		

Zetadocs Send Settings

## GetContactForReport

As explained in the previous section, not specifying a **Contact No. Field No.** means that this function will be called for you to return the value. This means that you are not restricted to extracting the reference directly from the

record being reported. The report id, record id and send settings are passed as parameters into this function and therefore you can implement your own custom logic for each report and record type to determine a contact. Note that the return value Code is not typed and is extra long. This means that the code is not tied to a specific NAV or BC type. This will be covered more in the next section.

### GetCompanyForReport

Similar to GetContactForReport this function allows you to implement custom logic for determining the **Company No.** value. The report id, record id and send settings are passed as parameters to this function and therefore you can implement your own custom logic for each report and record type to determine a company. Note that the return value Code is not typed and is extra long. This means that the code is not tied to a specific NAV or BC type. This will be covered more in the next section.

### GetHardCopyPrinter

This function is always called to get the printer to use for the current user when the delivery method is set to Hard Copy. It defaults to use the user's system default printer. It is provided here to allow you to change this logic.

### GetHardCopyPrinterForServer

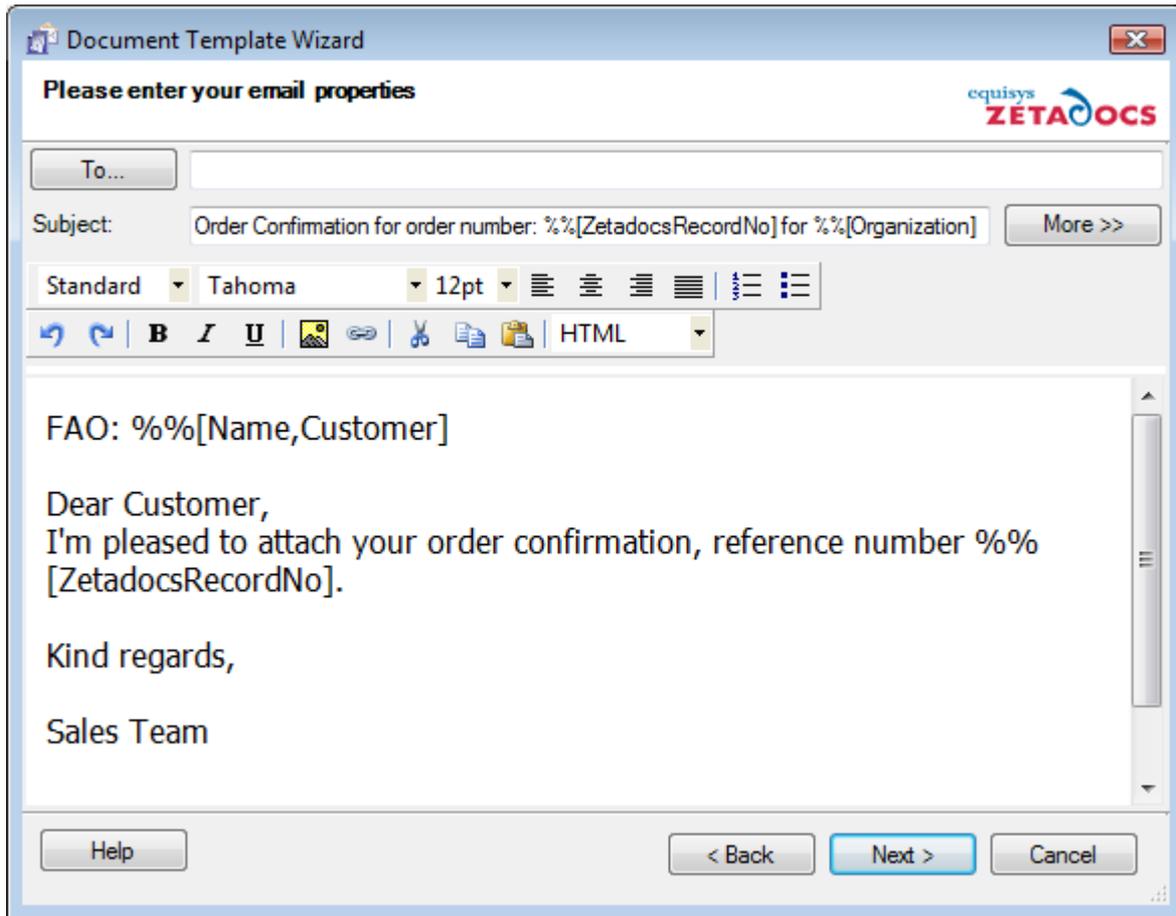
This function is always called before the server side delivery dialog. It can be used to override which printer the server uses. If this function returns an empty string, i.e. "", then the default printer for that report will be chosen following NAV or BC rules for the UseSystemPrinter report setting and the Printer Selections entries. If the function returns a non-empty string then that printer will be used for server side Hard Copy delivery. Multiple printers can be specified by returning a semi-colon separated string, e.g. 'Zetadocs PDF; Printer2'.

### GetOutputFileName

When a report is printed to Zetadocs, converted to a pdf and sent to a recipient it is desirable to be able to name that pdf file something relevant. This function is called from the AddRecord function in the Zetadocs Send Results Gen codeunit (i.e. once per report record) to allow you to specify independent file names.

### GetRecordNoForReport

Similar to the Company No. Field No. and Contact No. Field No. settings in the Zetadocs Report Settings there is an option to specify the Record No. Field No. This field's purpose is primarily for customizing Zetadocs message content. For example it is often desirable to include a Ref. number in the subject of email correspondence. This can be set by adding %[%ZetadocsRecordNo] into the subject of a Zetadocs Template. Once again this field no. reference setting operates similar to Contact No. Field No. such that if the field is left blank in the Zetadocs Report Settings then this function will be called from the **AddRecord** function in the Zetadocs Send Results Gen codeunit. This may be necessary if your reference number is composed of more than one field value. If so these field values could be concatenated and returned from this function.



Zetadocs Template Wizard

### GetCompanyRule

This function is exposed here for a number of reasons. Firstly it allows the customization of rules dynamically at runtime but primarily it can be used in conjunction with the Company Type Code to provide company rules for bespoke company types (i.e. types other than Customer and Vendor). This is explained in more detail in the next section. The important thing to note about this function is the parameter that is passed. The “Zetadocs Company Rule” is another temporary table maintained in the Zetadocs Send Results Gen codeunit. This is used to get the rule override information irrespective of the company type. This will become clearer in the next section. It is recommended that you review the implementation of this trigger function in the Zetadocs-Send Integration codeunit to further your understanding.

### OnAfterZetadocsSend

This function allows to execute actions after a successful delivery job using Zetadocs Server. This function can be used to edit a field in a record or even post the record after sending it. This function is called after the Zetadocs Send and Archive and it executes once per record to be sent in a batch. Examples can be found on the Zetadocs Integration codeunit.

## 2.3 Custom Addressing

Due to the nature of NAV or BC system installations and the hugely customizable potential it is acceptable that some systems may or may not use the company tables **18 Customer** and **23 Vendor**. Similarly some systems may not use the Contact Management Module or the native NAV or BC **5050 Contact** table. These entities may have bespoke tables, this section details how Zetadocs can be integrated into such systems. This section focuses on the abstract relationship between the “Zetadocs Company Types” table and the tables that represent these types. We will also see how Zetadocs can be extended to integrate with other existing native company types.

### Company Types

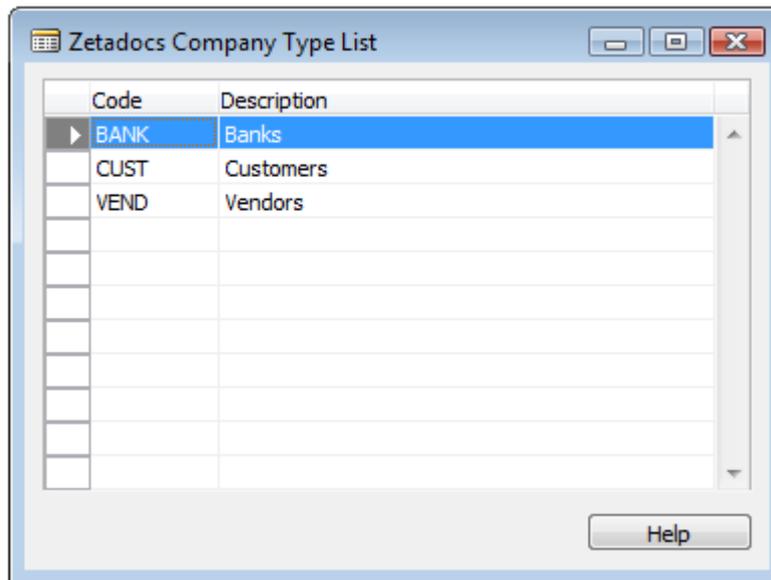
Zetadocs integrates with two native company types in its initial configuration, Customer and Vendor. However, the code which directly accesses these tables is kept in 2 locations that are public to a NAV or BC VAR. As an example of how one could implement a new bespoke company type or in fact edit one of the existing company types CUST and VEND the following example is presented.

NOTE: the Zetadocs Sales Document Queue form (9009962) operates for the Sales process only and uses direct references to the Customer and Contact names field in the **Sales Header** table. If you have modified the Sales Header table (36) to work with your bespoke Customer and/or Contact implementations then you may need to alter the references to **Sell-to Customer...** and **Sell-to Contact...** field references.

### BANK Example

Consider the business relation BANK. To integrate Zetadocs with reports that are sent to BANK contacts you would follow these instructions.

- Create a Zetadocs Company Type BANK.



Zetadocs Company Types

- Implement the BANK IF case to get the bank name and addressing information from your bank table in the Zetadocs Send Result table (9009968) similar to the CUST case.

```

Company No. - OnValidate()

IF "Company No." <> '' THEN
    BEGIN
        //"Company Name" := ZdIntegration.GetNameFromCompany("Company Type", "Company No.");
        IF "Company Type Code" = 'CUST' THEN
            BEGIN
                IF Cust.GET("Company No.") THEN
                    BEGIN
                        "Company Name" := Cust.Name;
                        IF "Contact No." = '' THEN
                            BEGIN
                                IF Cont.GET(Cust."Primary Contact No.") THEN
                                    BEGIN
                                        "E-Mail" := Cont."E-Mail";
                                        "Fax No." := Cont."Fax No.";
                                        IF Cont.Type = Cont.Type::Company THEN
                                            "Contact Name" := Cust.Contact
                                        ELSE
                                            "Contact Name" := Cont.Name;
                                        END
                                    END
                                ELSE
                                    BEGIN
                                        "E-Mail" := Cust."E-Mail";
                                        "Fax No." := Cust."Fax No.";
                                        "Contact Name" := Cust.Contact;
                                    END;
                                END;
                            END
                        END
                    ELSE
                        BEGIN
                            MESSAGE(msgCompanyNotFound);
                            zdUtilities.Log(2, STRSUBSTNO('T9009972 - Company No. - OnValidate: Customer %',
                                zdUtilities.Log(2, '          Please check that "Company No. Field No." is correct
                            END;
                        END
                    END
                ELSE IF "Company Type Code" = 'VEND' THEN

```

Zetadocs Send Result Table Code

- Also update the Company No. - OnLookup trigger and Contact No. - OnLookup trigger to include the IF case for your BANK type.
- Update the tbContName – OnLookup trigger in the Zetadocs Delivery Form (9009966) to use the correct ContactBusinessRelation information if necessary.

Zetadocs Delivery Form “tbContName” textbox

- Then in the Zetadocs Report Settings set the BANK Company Type for the report that has BANK recipients and set the respective Company No. Field No. and Contact No. Field No. references for the report source table.
- If you have implemented a bespoke Customer table for use with Sales and Marketing then you may need to update the Zetadocs Sales Document Queue Form (9009962) to use your bespoke Customer record type references where necessary.

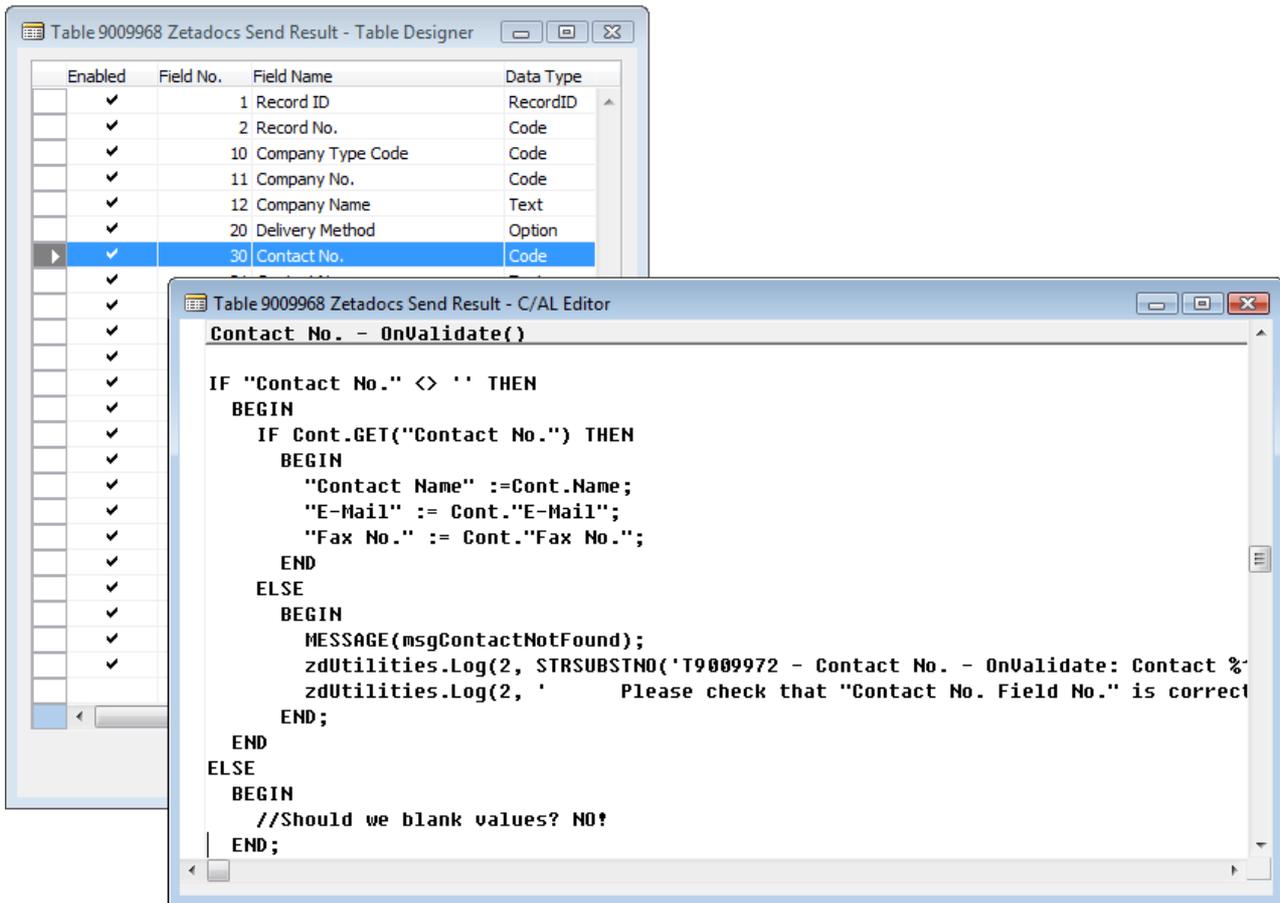
## Contacts

Similar to Company it is possible that there may be a bespoke implementation of Contact information or there may be a requirement to change the default contact address resolution logic of Zetadocs.

### Validating

As has been explained the Zetadocs Report Settings the **Contact No. Field No.** determines whether the contact reference can be extracted from a field in the record that is passed to the Zetadocs Send Results Gen codeunit or whether this is implemented by the VAR and called from the codeunit. Once the reference has been set in the **Contact No.** field in the Zetadocs Send Result table the OnValidate trigger is called. Therefore to modify the contact addressing resolution or to change the code to use a bespoke contact table you must follow these instructions:

- Modify the Contact No. – OnValidate trigger in the Zetadocs Send Result table (9009968) to get the name and addressing information from your bespoke contact.



Zetadocs Send Result table - Contact No. code

- Modify the Contact No. – OnLookup trigger in the Zetadocs Send Result table (9009968) to use the correct ContactBusinessRelation information dependant on the Company Type Code.
- Update the Zetadocs Delivery Form (9009966) references to contact information to use your bespoke contact record type and correct ContactBusinessRelation information if necessary.

## 2.4 Custom Delivery Options

By now you should have a good idea of what this codeunit is for. Up until now we have been exploring the internal processing and control flow of the Zetadocs Addin. By using specific settings and writing some code in the Zetadocs-Send Integration codeunit and the Zetadocs Send Result table triggers it is possible to completely integrate and customize Zetadocs into an NAV or BC system. We have also seen the potential to implement business specific logic into the sending mechanism and expand on the Zetadocs product functionality. There is still however room to take it further.

This section details the two basic functions in the Zetadocs-Send Customize codeunit (9009962) that allow the implementation of more business specific send logic as well as document formatting and archiving customizations.

### **GetAdditionalEmbComms**

This function is deprecated in version 5.0 of Zetadocs. It remains for backward compatibility with the old report mark-up method. Please use `WriteAdditionalEmbCommsToFile` instead.

### **OverrideSendResult**

As the name suggests this function gives you the whole send result. At this point the result has been completely resolved by Zetadocs and any custom code that has been implemented by you in the various areas explained earlier in this document. This then allows you to modify the completed record as a whole. The only place the results can be edited after this are in the Zetadocs Delivery Dialogs or the Zetadocs Errors Dialogs depending on your display settings for these dialogs, as specified in the Zetadocs Report Settings (potentially overridden by the `OverrideSendSettings` in the Zetadocs-Send Integration codeunit).

Let's examine the Zetadocs Send Result table a little more closely now for the first time.

Enabled	Field No.	Field Name	Data Type	Length	Description
<input checked="" type="checkbox"/>	1	Record ID	RecordID		The record to which the result refers
<input checked="" type="checkbox"/>	2	Record No.	Code	80	Optional storage of a record no. ref for archiving as metadata
<input checked="" type="checkbox"/>	10	Company Type Code	Code	20	Type of Company
<input checked="" type="checkbox"/>	11	Company No.	Code	80	
<input checked="" type="checkbox"/>	12	Company Name	Text	50	Company Name
<input checked="" type="checkbox"/>	20	Delivery Method	Option		The delivery addressing to use in Zetadocs (Email, fax, Print)
<input checked="" type="checkbox"/>	30	Contact No.	Code	80	Reference to the Contact specified on the order or whatever we are s...
<input checked="" type="checkbox"/>	31	Contact Name	Text	50	Contact Name
<input checked="" type="checkbox"/>	40	E-Mail	Text	250	The Email To address line
<input checked="" type="checkbox"/>	41	E-MailCc	Text	250	Customizable Cc addressing
<input checked="" type="checkbox"/>	42	E-MailBcc	Text	250	Customizable Bcc addressing
<input checked="" type="checkbox"/>	43	E-MailFrom	Text	80	The Email From address line
<input checked="" type="checkbox"/>	50	Fax No.	Text	250	
<input checked="" type="checkbox"/>	60	Printer	Text	250	
<input checked="" type="checkbox"/>	70	ArchivingEnabled	Boolean		
<input checked="" type="checkbox"/>	71	Archive Location	Text	250	The root sharepoint archive folder
<input checked="" type="checkbox"/>	72	Archive Document Type	Text	50	
<input checked="" type="checkbox"/>	80	Zetadocs Archive ID	GUID		
<input checked="" type="checkbox"/>	85	Zetadocs Record Link	Text	250	The Zetadocs Record Link for this record
<input checked="" type="checkbox"/>	86	Zetadocs Record Link Type	Text	10	The Type of record link
<input checked="" type="checkbox"/>	90	Zd Template ID	Code	20	The Zetadocs Template to use, from Zetadocs Template table
<input checked="" type="checkbox"/>	91	Zd Template Name	Text	100	Name from Template table.
<input checked="" type="checkbox"/>	110	Error	Boolean		Error flag
<input checked="" type="checkbox"/>	111	Error Description	Text	250	
<input checked="" type="checkbox"/>	112	Skip	Boolean		Should the record be skipped in the batch
<input checked="" type="checkbox"/>	130	Output File Name	Text	150	Customizable output name of pdf sent
<input checked="" type="checkbox"/>	140	Lines No.	GUID		Used to references other results tables in a header lines setup e.g. ad...
<input checked="" type="checkbox"/>	150	Record ID String	Text	250	The record to which the result refers
<input checked="" type="checkbox"/>	160	NetworkArchivingEnabled	Boolean		
<input checked="" type="checkbox"/>	161	Network Archive Location	Text	250	The root network archive folder
<input checked="" type="checkbox"/>	180	ID	GUID		
<input checked="" type="checkbox"/>	190	XML	BLOB		
<input checked="" type="checkbox"/>	200	Batch ID	GUID		
<input checked="" type="checkbox"/>	210	Delivery Status	Text	250	
<input checked="" type="checkbox"/>	220	Additional E-Mail	BLOB		
<input checked="" type="checkbox"/>	230	Additional E-MailCc	BLOB		
<input checked="" type="checkbox"/>	240	Additional E-MailBcc	BLOB		
<input checked="" type="checkbox"/>	250	Group	GUID		
<input checked="" type="checkbox"/>	300	Dynamic Commands	BLOB		

Zetadocs Send Result

Some of these fields will be familiar from having worked with the Zetadocs Company Rule used in the GetCompanyRule trigger function in the Zetadocs-Send Integration codeunit. Let’s break them up into their respective parts and get familiar with their use and how we can exploit this to implement our specific business logic and workflow.

Record Fields (1 – 2)

The **Record ID** and **Record No.** represent the typeless and typed references to the record we are sending. Record ID is used by the core processing of Zetadocs to tie the send results and records together. Record No. is what will ultimately be passed to Zetadocs for use as an external reference by either the recipient or in archiving.

**Example Usage:**

Having the Record ID of what you are sending means you can get that record and query or use values from that record to implement logic for changing other fields in the Zetadocs Send Result.

**Company Fields (10 – 12)**

These fields and their use should be extremely clear by now. The **Company Type Code** allows us to integrate with native and bespoke company tables such as Customer, Vendor and one of your own design. **Company No.** and **Company Name** are then obvious references to the id and display name of the company.

**Delivery Method (20)**

This controls the addressing that is selected for sending, E-Mail, Fax or Hard Copy. Some obvious customization possibilities include implementing a company policy such as “Any company sensitive information should not be emailed under any circumstance and should be faxed or printed only.” Depending on particular reports or recipients the delivery method could be checked or changed to uphold company policy.

**Contact Fields (30 – 31)**

In collaboration with the Company fields these fields can integrate with native and bespoke systems for contact addressing information resolution. By setting and validating this field it is possible to pull the information from the contact based on the logic that is implemented in the Zetadocs Send Result table triggers as discussed previously.

**Addressing Fields (40 – 60)**

Provides the ability to set additional email recipients in the Cc and Bcc fields, these fields are without relationships to the contact and company references that may or may not be present in the send result record. This means that addressing information can be taken from anywhere for the respective email, cc, bcc, fax and print addressing fields. Care should be taken to clear the company and contact no. fields before setting these values manually.

**Archiving Fields (70 – 72)**

Set in the Zetadocs General Settings, archiving settings are available for per record modification. This can be used for example in deciding whether or not the document is archived depending on its content or changing the archiving location to a limited permissions document library in the archiving site, or sub categorizing documents using the Zetadocs Archive Document Type.

**Properties:**

- Archiving Enabled – per document archiving
- Archive Location – the archive connection string. The example shipped in the OverrideSendResult function details how to change the subfoldering plan for different records. The subfolder string should be of the format '{Archiving}: {ArchiveLocation},{subfolderPath}', where {Archiving} is sharepoint or zetadocs, {ArchiveLocation} is the archiving site, and {subfolderPath} is the path where the file is going to be stored. Please review the example if you require further information.
- Archive Document Type

**WARNING:** Consideration should also be taken when modifying the subfoldering of documents to SharePoint as this can dramatically affect performance or visibility of document from SharePoint in NAV or BC. Please refer to the advice on file planning in this guide.

### Zetadocs Archive ID (80)

This field has been deprecated in version 5.0 of Zetadocs and should be ignored. It remains for backward compatibility. For upgrade systems, please read the warning below.

**WARNING:** This field will contain data which contains a reference for an NAV or BC record to archived documents. **It should not be edited or you risk breaking the archive search functionality of Zetadocs.**

### Zetadocs Record Link (85)

**WARNING:** This field will contain data which contains a reference for an NAV or BC record to archived documents. **It should not be edited or you risk breaking the archive search functionality of Zetadocs.**

### Zetadocs Record Link Type (86)

**WARNING:** This field will contain data which contains a reference for an NAV or BC record to archived documents. **It should not be edited or you risk breaking the archive search functionality of Zetadocs.**

### Zetadocs Template Fields (90-91)

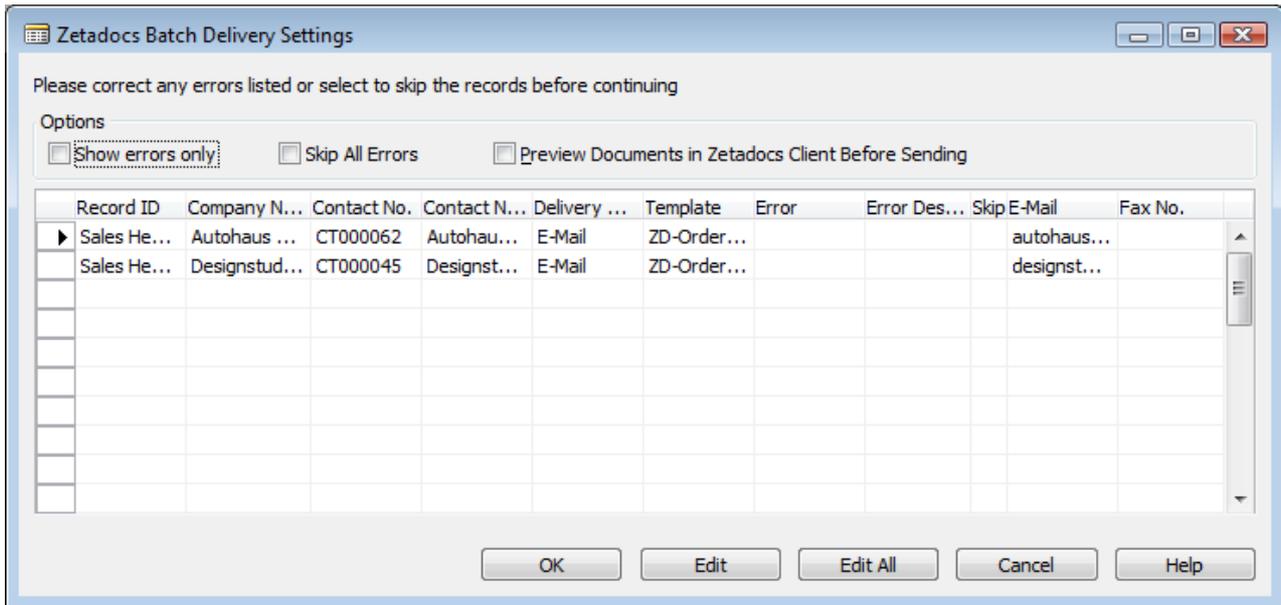
The “Zd Template Name” field is set automatically as the “Zd Template ID” field has a table relation with the Zetadocs Template table. The uses of this field are fairly obvious, change the Zetadocs template being used depending on some custom business logic.

### Error Fields (110-111)

The Zetadocs Send Result table (9009968) has a trigger function **CheckErrors**. This function is called for each results record in the Zetadocs Send codeunit during the **ProcessRecords** function. This happens after all possible modifications of the send result values have taken place in the Zetadocs-Send Integration trigger functions and Zetadocs-Send Customize codeunit override function **OverrideSendResult**. This function checks the addressing information (email, fax, and printer) against the specified Delivery Method (Email, Fax, Hard Copy) to pre-empt potential send failures in the Zetadocs Client. If an error is found in a Zetadocs Send Result record the **Error** field is set to TRUE and a description of the error written in **Error description**. It is possible to implement your own error checks by editing the code in the **CheckErrors** function in the Zetadocs Send Result table. Note that the presence of an just 1 error record in the Zetadocs Send Result record for a particular send will cause the Zetadocs Batch Delivery Dialog to display **IF** the **Zetadocs Send Settings** for that send has **Show Batch Delivery Dialog** set to TRUE otherwise all errors will be marked as Skip = TRUE.

### Skip (112)

When Skip = TRUE this will cause the report to use CurrReport.Skip for the associated record. For example if your report contains 3 reports and one is marked to skip then only 2 will be printed to Zetadocs. If the Zetadocs Batch Delivery Dialog is displayed either by there being errors in the send results or by the user choosing to preview the send results, they can manually set a report to skip particular report records.



Zetadocs Batch Delivery Dialog

### Output File Name (130)

When a report is printed to Zetadocs and split into the addressable documents each document is given a name. It is desirable to be able to control the name of these documents as it will be the name of the PDF file produced and attached to emails as well as archived. This field allows you to set that document name. A good example of its use would be to generate the name based on the type of report and the record no. For example: "Order Confirmation: 10045". Using the RecordID you can retrieve any information about the record for generating the output file name.

### Lines No. (140)

WARNING: This field will contain data used internally by the Zetadocs core processing.

**It should not be edited or you risk breaking functionality of Zetadocs.**

### Record ID String (150)

WARNING: This field will contain data used internally by the Zetadocs core processing.

**It should not be edited or you risk breaking functionality of Zetadocs.**

### Network Archiving Fields (160 - 161)

Set in the Zetadocs Delivery Settings, the archiving settings are available for per record modification. This can be used for example in deciding whether or not the document is archived depending on its content or changing the archiving location.

#### Properties:

- Network Archiving Enabled – per document archiving
- Network Archive Location – the archive connection string is the desired network archive path and takes the following format

“//\*yourmachinename\*/\*yoursharedfolder\*/[subfolder[/subfolder...]]”

### ID Field (180)

Primary key of the table, please ignored this field as it shouldn't be changed manually.

### Zetadocs Server delivery Fields (190 - 300)

This fields are internal to the server delivery, please ignored these fields as they shouldn't be changed manually.

### WriteAdditionalEmbCommsToFile

In the final stage of the reporting process for Zetadocs the **GetRecordCustComms** function is called from the report on the Zetadocs Send Results Gen codeunit to retrieve the formatted embedded commands which represent the send result for each report record. Internally in the Zetadocs Send Results Gen codeunit we create the Zetadocs embedded commands but before we return it we call this trigger function

**WriteAdditionalEmbCommsToFile** in the Zetadocs-Send Customize codeunit. This allows you to return additional embedded commands which can be used to customize your document message subject and body text as well as additional archive metadata to associate with the document when it is archived.

### Dynamic Fields

Dynamic Fields are name/value pairs that can be used to pass any information to Zetadocs. The format of the command is:

```
%%[field: <name>, <value>]
```

For Example:

```
%%[field: salutation, Mr.]
```

By using this in conjunction with a Zetadocs Templates you can personalize the message as follows.

```
%%[field: salutation, Mr.]
```

```
%%[field:name, McCarthy]
```

```
%%[field: company, Widgets Inc.]
```

```
%%[field: quote_number, 1234]
```

```
%%[field: myname, Sam]
```

### Document Template

When merged to a document template containing the following text:

```
Dear %%[salutation] %%[name]
```

```
I'm pleased to attach your quotation as discussed along with our conditions of sale and product brochure.
```

```
Our quotation number for your reference is %%[quote_number], please include this on your sales order.
```

```
Regards,
```

```
%%[myname]
```

### Expected Output

Zetadocs creates the following email message body:

```
Dear Mr. McCarthy
```

```
I'm pleased to attach your quotation as discussed along with our conditions of sale and product brochure.
```

```
Our quotation number for your reference is 1234, please include this on your sales order.
```

```
Regards,
```

```
Sam
```

## Archiving Metadata

**Please note:** this feature is not available when using the Zetadocs Archive.

It is also possible to archive any of the dynamic fields as metadata. To do this the embedded command should include the ARCHIVE flag at the end. Here is the format:

```
%%[field: <name>, <value>, ARCHIVE]
```

For Example:

```
%%[field: Postcode, N7 0JE, ARCHIVE]
```

Whether the dynamic field is used in personalizing the message or not its value name pair will be passed to the SharePoint archive as a content type column.

There are a couple of things to note about this. If the column name does not exist already in SharePoint it will need to be created as a column of the content type Zetadocs or of a content type that is a sub type of the Zetadocs content type, this needs to be done in SharePoint in advance.

Zetadocs documents default to using a Zetadocs content type. If you have specified an Archive Document Type then the content type must be a child type of the Zetadocs content type. NOTE: It is essential that the SharePoint columns and content types are created before archiving a document and that it is well tested before roll out.

## Add a File attachment

### Zetadocs Client Delivery

Allows you to add a file to the email being sent by Zetadocs, this file will not be converted to pdf and merged with the pdf output but will instead be included with the email as an additional attachment.

```
%%[fileattachment: «Filename»]
```

Where «Filename» is the file to include located in a public, private folder or even full URL.

Examples:

```
%%[fileattachment: C:\Users\Public\Documents\SalesInvoice.docx]
```

```
%%[fileattachment: http://www.equisys.com/img/equisys\_logo.gif]
```

### Extended with Zetadocs Server Delivery

When used with Zetadocs Server Delivery, the file attachment Dynamic Command can be extended to merge a stationery to the document. For this operation, both the document and stationery should be pdf files. The format is:

```
%%[fileattachment: «Filename», «Stationeryname»]
```

Where «Filename» is the file to include located in a public, private folder or even full URL.

Where «Stationeryname» is the stationery file located in a public, private folder or even full URL. This parameter is optional.

Examples:

```
%%[fileattachment: C:\Users\Public\Documents\SalesInvoice.pdf, \\localhost\Zetadocs  
Templates\CompanyStationery.pdf]
```

## 2.5 Delivery Automation

You can automate and schedule the email delivery of NAV or BC reports with Zetadocs Delivery Plus. The example provided below describes how to create a new NAV or BC Codeunit that calls the Zetadocs Server Send Codeunit to send a report automatically. You can modify the example to suit your requirements.

### Assumption

These steps assume that Zetadocs is correctly setup and configured to use a Zetadocs Delivery Plus license, and have been written for a NAV 2015 system. Similar steps may apply to later versions of NAV or BC.

### Prerequisite

Zetadocs needs access to stored credentials in order to archive files to your SharePoint site.

#### If you are using the Document FactBox in the NAV or BC Web client

1. Create a test file.
2. Open the NAV or BC Web Client as the user that will run the automation.
3. Open the Document FactBox in any modified page, e.g. Sales Invoice page.
4. If your credentials are not in the server, Zetadocs will display a dialog. Insert your credentials.
5. Delete the test file from your Archiving site; to do so, delete the test file in the FactBox.

#### If you are using the Document FactBox in the Dynamics NAV Client

1. Open the Dynamics NAV Client as the user that will run the automation.
2. Open a page that contains the Zetadocs Send button.
3. If your credentials are not in the server, Zetadocs will display a dialog. Insert your credentials.
4. After inserting the credentials, please click cancel to stop the process.

### Procedure

#### Step 1: Create a Codeunit

1. Open the Development Environment and open the Object Designer. Select the Codeunit tab.
2. Create a new Codeunit: File -> New.
3. Save and Compile the new Codeunit, type a name and an ID, e.g. Codeunit Name: Automate Report Delivery, ID: 51002.

#### Step 2: Call Zetadocs Server Send

1. Open the Microsoft Development Environment and open the Object Designer. Select the Codeunit tab.

2. Select the Codeunit created in the previous step and click on Design.
3. Create a new public function:
  - a. Click View->C/AL Globals
  - b. Select the Functions tab.
  - c. Insert the name of the function, e.g. SendOrderConfirmationReport.
  - d. Go to the properties of the function (View->Properties) and select "Local: No".
  - e. Close the Properties dialog.
  - f. In the Functions tab of the C/AL Globals dialog, select the function you have added (e.g. SendOrderConfirmationReport) and click on Locals.
  - g. Add a new Parameter:
    - Name: SalesHeaderNoFilter
    - DataType: Text
4. Create new Local Variables:
  - a. Select the Variables tab.
  - b. Add the following variables:
    - Name:Parameters
    - DataType:Text
  
    - Name:FilterTemp
    - DataType:Text
  
    - Name:ZdServerSend
    - DataType:Codeunit
    - Subtype:Zetadocs-Server Send
5. Close the dialog screens and go back to the one that contains the code.
6. Scroll to the body of the function created previously and paste the following code:

```
//Initialize the Filter
FilterTemp := SalesHeaderNoFilter;
IF FilterTemp = '' THEN
BEGIN
  //If the filter is empty, get all the items
  FilterTemp := '*';
END;

//Create the parameters needed for the Order Confirmation.
//Please consult REPORT.RUNREQUESTPAGE from the NAV or BC help for more information.
Parameters := '<?xml version="1.0" standalone="yes"?>'
+'<ReportParameters name="Order Confirmation" id="205">'
+'<Options>'
+'<Field name="NoOfCopies">0</Field>'
+'<Field name="ShowInternalInfo">>false</Field>'
+'<Field name="ArchiveDocument">>false</Field>'
+'<Field name="LogInteraction">>true</Field>'
+'<Field name="DisplayAssemblyInformation">>false</Field>'
+'</Options>'
+'<DataItems>'
+'<DataItem name="Sales Header">SORTING(Document Type,No.)
WHERE(No.=FILTER('+FilterTemp+'))</DataItem>'
```

```
+ '</DataItems>'
+ '</ReportParameters>';

//Submit the batch, Zetadocs will process it in the background
ZdServerSend.SendServerDelivery(205, Parameters, FALSE);
//Where SendServerDelivery(ReportID : Integer;RequestPageParams : Text;ShowUI : Boolean)
//Please note that ShowUI must be FALSE for automation, and it only works if the
RequestPageParam is not empty.
```

7. Close the dialogs, save and compile the Codeunit.

**Note:** You can easily extend this example to get the parameters from the REPORT.RUNREQUESTPAGE and store the result the BLOB field of a NAV or BC table. This means, you could get the parameters from the table and pass them to Zetadocs instead of hard-coding them in the codeunit.

### Step 3: Test your system

In order to test this example, you just need to call the function as follows:

```
SendOrderConfirmationReport(<SalesHeaderNoFilter>);
```

Where <SalesHeaderNoFilter> is the filter that will be applied for the Sales Header No. (e.g. SO20001).

Please ensure that everything works as expected before deploying it live.

### Tip

An easy way to test your system is to add the previous call to the OnRun trigger of the Codeunit that has been created in the previous step. You would save and compile the Codeunit, and then run it from the Development Environment.

There are alternative ways to test this example depending on your needs or requirements. For example, you can call this Codeunit from the NAV or BC Job Queue or you may expose the Codeunit as a NAV or BC Web Service.

### Notes

- If you are using NAV or BC Job Queues, you may need to repeat the Prerequisite. You may also need to login as the user that is running the NAV or BC Job Queue.
- Please remember that NAV or BC Job Queues run the OnRun trigger. This means you may need to add a call to SendOrderConfirmationReport in this trigger.



## 3. Zetadocs Capture Plus

**Note:** The following sections of the SDK guide cover the setup of features provided as part of the Zetadocs Capture Plus module. These changes can only be made by those with the Zetadocs Capture Plus SDK which requires Capture Plus module licence from Equisys.

The Zetadocs Capture Plus is an optional component of Zetadocs and is required for implementing custom Document Queues and bespoke business logic in Zetadocs capture features. At its core is the Zetadocs Server which runs on your server and converts documents placed into a Shared Network folder into PDF/A format. It works with NAV or BC and enables the document queues to work with a wide range of documents by converting inbound documents into PDF/A. As standard the document queue supports a variety of file types, allowing viewing, processing and eventual archive of these documents.

This provides a number of benefits:

- Documents are stored in a standard storage format (PDF/A)
- Document content is searchable within SharePoint including text which is part of graphics, if OCR is enabled (optional requires additional licence)

By storing all documents within SharePoint you enable a wider range of your company's staff to access these documents despite perhaps not having a copy of the native software for the document. It also means that you have a copy of the document that is an accurate representation of the document when it was received.

The list below covers some of the options available to Capture Plus customers, typically these are used to streamline document handling and management processes.

### **Capture Plus Features Include:**

- Capture and Conversion of inbound documents to PDF/A format
- Custom Document Queues
- Optical Character Recognition
- Barcode Recognition
- On screen approval with workflows
- Document Association
- Bespoke business logic and capture feature customizations

### 3.1 Understanding the Capture Plus SDK Framework

Before continuing it is recommended that you read the [Zetadocs SDK introduction](#) section of this document.

The three elements to the Zetadocs Capture Plus SDK are:

1. Interfaces

- Zetadocs Document Queue (Form / Page)
  - This has been designed to work for any queue and should not be modified to implement customizations without consulting Equisys Ltd.
- NAS
  - It is possible to author a NAS service to automatically perform scheduled document queue processing.

2. Zetadocs Core Codeunits

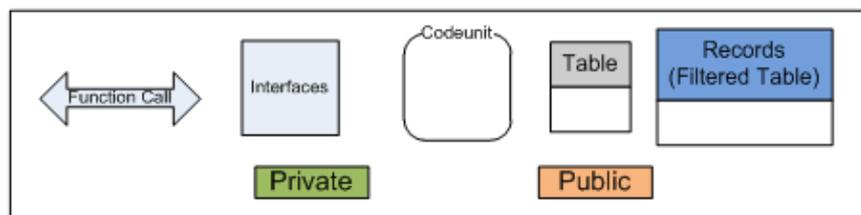
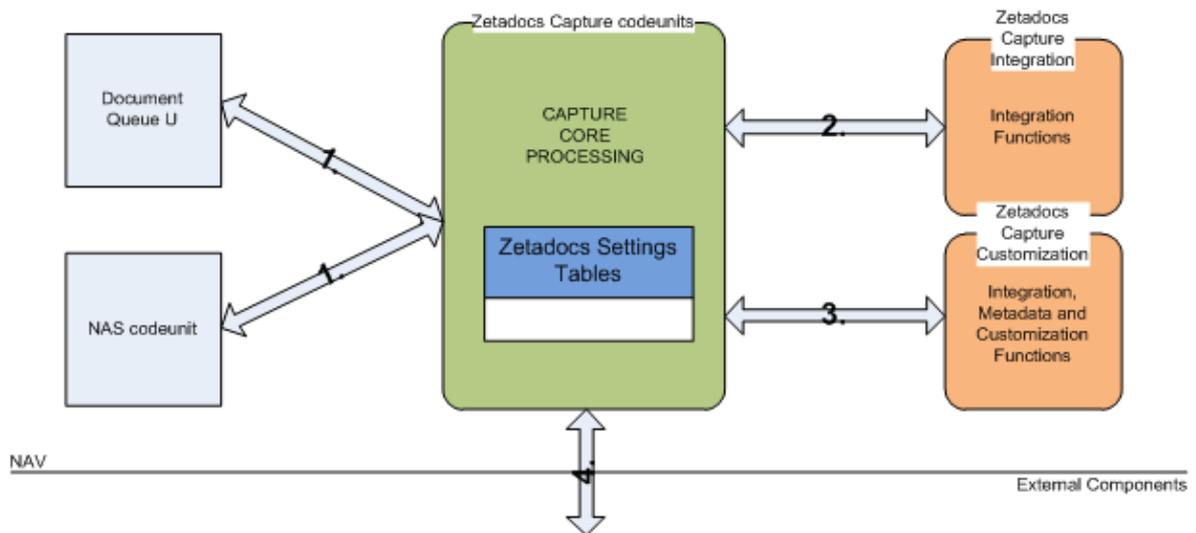
- Zetadocs Capture codeunit
  - This codeunit provides functions that are called by the Interfaces to complete linking actions.

3. Integration and Customization Codeunits

- Zetadocs Capture Integration codeunit
  - This is currently not used and is only present for backward compatibility with earlier versions
- Zetadocs Capture Customize codeunit
  - This codeunit has functions called by the Zetadocs Core codeunits to allow the developer to implement custom queues and bespoke business logic.

**Process Flow**

The processing flow follows the standard Zetadocs SDK pattern.



Notation

Zetadocs Capture Plus Process Flow

## Capture Plus Example Scenario

It is easier to understand how these features all come together by looking at a brief example of how they can be used to improve document handling.

Scenario: A company sends out Shipment notes printed from NAV or BC with their deliveries, these are signed by the customer and returned before being processed and filed.

Solution: By using Capture Plus you can automate and streamline this process significantly, firstly on printing the Shipment note from NAV or BC, Zetadocs can add a Barcode to the document. It is then signed as usual by the customer but on return it is simply scanned. Zetadocs can then recognise the barcode and automatically adds a Pdf copy of the document to your archive and associates it with the original order that the Shipment note was based upon. This is done using a custom Shipment Document Queue. It gives the user processing these documents a single location from where they can view the signed Shipment Note and create and send the corresponding invoice. This provides the organisation with an archive that is populated with related records e.g. a Quote, Order, Shipment Note, Returned Signed Shipment Note, Invoice which enables them to view the whole transaction workflow from the archive. We will demonstrate how to setup this Shipment example in the coming pages.

The sample code to achieve this is shipped as part of the Zetadocs-Capture Customize codeunit (9009964).

## 3.2 The Zetadocs Server

### How the Zetadocs Server works

#### Conversion

Documents are added to the document queue by placing them in a shared network folder. These are then displayed within the document queue allowing the user to process the document and create associated entries and documents. The Zetadocs Server application automatically converts documents placed within the shared network folder into PDF/A whilst storing a copy of the original document in another folder.

#### PDF Format

The Zetadocs Server converts documents in supported formats into **PDF 1.4 (PDF/A-1a)** format. This is an international ISO standard ISO 19005-1:2005 for more details on this format please see ([http://www.pdfa.org/doku.php?id=pdfa:en:pdfa\\_whitepaper](http://www.pdfa.org/doku.php?id=pdfa:en:pdfa_whitepaper)).

#### OCR

If you choose to include OCR as part of your solution all documents converted to PDF will support full text search. This includes graphics documents or documents that contain graphics (e.g. a PDF with embedded graphics), this requires special processing. To provide this we have the ZD OCR Processor, this is an optional module which incorporates the Abbyy FineReader Engine. The full text that is recognised is added to the created PDF file as a text layer, this enables full text search.

### Supported Inbound File Formats

The following file formats are supported, how they are processed depends on whether the ZD OCR Processor has been included:

If the OCR module is installed then the following file types will be processed with OCR:

- PDF (.pdf)
- BMP and DIB (.bmp, .dib)
- PCX and DCX (.pcx, .dcx)
- JPEG, JIFF (.jpg, .jpeg, .jpe, .jfif)
- GIF (.gif)
- TIFF (.tif, .tiff, .g3n, .g3f)
- DjVu (.djvu, .djv)

If the ZD OCR Processor is not included files will be passed for conversion without being OCR'd, we support the following file types:

- Office files (v2003 and v2007) – Word and Excel (.doc, .docx, .xls, .xlsx)
- Rich text files (.rtf)
- Web pages (.html, .htm)
- Text files (.txt)
- Email messages (.msg format is recommended as htm format doesn't include attachments)

Email message conversion of .msg files results in conversion of the email message body text and the unpacking and conversion of any supported attachments. Should any of the attachments not be in a supported format then the whole message will be flagged as having failed to convert so the user can review it in its original format.

The following file types are not supported:

- Compression: Zip, Rar, ARJ, TAR, TGZ, CAB ...
- Media: MP3, WMV, AVI, MPG, MPEG, MP4, WAV...
- Executable types: EXE, DLL, BAT, VBS ...

### Multipart Files

The Document Converter supports multipart files e.g. an email with an attached purchase order, this is then converted into a pdf file with two or more pages with bookmarks to indicate where the various elements of the message are contained. In the above example you would be presented with a first page containing the email subject line, body text, addressing information and email message.

- The following Email metadata is also retained:
- Sent – time/date
- Received – time/date
- From
- To
- CC
- Priority

Next would follow the attachments, each with its own separate bookmark within the pdf.

**Note:** Hosting the document queue folders on a separate machine to the one the Zetadocs Server is running on is not supported.

## Batch Scan

The Zetadocs Server supports batch scan splitting from a network scanner. It does this by monitoring a Batch Split subfolder in each of the configured document queues and processing any documents which are deposited in it for splitting. As such you will need to configure settings on your network scanner to drop scanned documents into the relevant folder e.g.:

**<DocumentQueuePath>\Batch Split**

The Zetadocs Server processes the batch and submits the split documents to your Document Queue using the specified splitting method, these are:

- Page Interval - The Zetadocs Server splits the batch of documents using a constant page interval which can be configured, e.g. resulting in the batch being split every 4 pages.
- Barcode - The Zetadocs Server reads the document and splits it on detecting a barcode. If you are using a barcode as a whole splitter page, if for example you were adding specific barcode pages to separate the batch, you may want to delete the split page. In this case the Zetadocs Server can be configured to discard the splitter page automatically.
- Barcode value change - If you are scanning batches with barcodes on each page, the Zetadocs Server splits the document each time the detected barcode value changes.

**Note:** The barcode batch splitting methods require Capture Plus and the Abby Software installed and configured. If you are scanning documents that contain several barcodes on the same page, a filtering prefix can be configured exclude all barcodes without a specific prefix.

## The Zetadocs Capture Plus Install

Zetadocs Capture Plus is based around the integration of the Zetadocs Server with NAV or BC and your other Zetadocs components to provide a range of enhanced functionality.

This guide will take you through the setup of each of the following:

- [Zetadocs Server](#)
- [Custom Document Queues](#)
- [Automation Using Barcodes](#)
- [Automation Using Extracted Text](#)

To do this we will demonstrate how to setup the Shipment Queue from the [earlier example](#).

## 3.3 Zetadocs Server Setup

The Zetadocs Server should already have been installed as part of a Capture Essentials system, as such the following sections will only detail those steps pertinent to enabling the Capture Plus features.

In this section we will be creating a Shared Network folder specifically for our custom Shipment Queue and configuring the Zetadocs Server to point at this folder. Please refer to the sections in the Essentials Installation Guide for instructions on how to install the Zetadocs Server. This guide will take you through adding a queue to the configuration either instead of or on top of the Sales and Purchase Queues.

## Configuring the Zetadocs Server

The Zetadocs server has a number of settings which allow you to specify how it handles documents, many of these will have been configured during the Capture Essentials setup earlier. We will cover those specific to Capture Plus here, which are:

1. [Creating a New Queue Entry](#)
2. [Enabling and Disabling Abbyy](#)
3. [Batch Scanning Settings](#)
4. [Barcode Settings](#)

Zetadocs Server settings are modified using a Config.xml file, this is located in the Zetadocs Server installation location usually C:\Program Files (x86)\Zetadocs Server\Document Converter. Open this file in a text editing program such as Notepad.exe and edit as per the instructions in this section.

The Config.xml file consists of specific instruction for each queue setup, in the example below there is a Sales and a Purchase Document Queue inside the <docsQueue></docsQueue> tags and the general Zetadocs Server settings below that. Once you have changed the options as required save the updated Config.xml file.

**WARNING:** The Zetadocs Server Config.xml file is case sensitive, therefore when setting the various options please ensure that you copy the exact casing shown here. Failing to do so may prevent the server from working as intended.

### Example Config.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <docsQueue>
    <DocsQ Description="Sales Document Queue" Location="//User specified queue
location\Sales Document Queue\">
      <EnablePDFTextExtractionToFile>false</EnablePDFTextExtractionToFile>
      <BatchScanSplitType>Barcode</BatchScanSplitType>
      <BatchScanPageInterval>1</BatchScanPageInterval>
      <BatchScanBarcodePrefix></BatchScanBarcodePrefix>
      <BatchScanBarcodeDeleteSplitPage>false</BatchScanBarcodeDeleteSplitPage>
      <FilenamePrefix></FilenamePrefix>
    </DocsQ>

    <DocsQ Description="Purchase Document Queue" Location="//User specified queue
location\Purchase Document Queue\">
      <EnablePDFTextExtractionToFile>false</EnablePDFTextExtractionToFile>
      <BatchScanSplitType>Off</BatchScanSplitType>
      <BatchScanPageInterval>1</BatchScanPageInterval>
      <BatchScanBarcodePrefix></BatchScanBarcodePrefix>
      <BatchScanBarcodeDeleteSplitPage>false</BatchScanBarcodeDeleteSplitPage>
      <FilenamePrefix></FilenamePrefix>
    </DocsQ>
```

```

</docsQueue>

<PDFQuality>Better</PDFQuality>
<EnableAbbyy>true</EnableAbbyy>
<StellentTimeOut>10</StellentTimeOut>
<StellentBypassPDF>true</StellentBypassPDF>
<LogLevel>Error</LogLevel>
<LogToFile>true</LogToFile>
<LogToEventLog>>false</LogToEventLog>
<EnableBarcodeDetection>true</EnableBarcodeDetection>
</Config>

```

### 3.3.1 Adding a New Document Queue

Items added into the Document Queue folders will be processed and converted to PDF/A format. Document Queue folders should be folders on the local machine where the Zetadocs Server is installed. The folder should then be shared on the network for Document Queue users to access.

#### Creating the Document Queue Folders

Each document queue requires a network folder into which captured documents are placed for processing and conversion, and to which all users of the queue have full access permissions.

- Create the required folder or folders in this case called Shipment Queue.
- View its Properties and ensure the Share this Folder radio button is checked.
- Select the Permissions button and assign the necessary permissions.

USERS: Share permissions with a minimum of Change and Read Permissions. Full Permissions are recommended.

SERVICE: The Zetadocs DIRMonitor Service must have access to the queue folders. To do this add FULL Permissions to the SYSTEM user or whatever network user the service is running as.

#### Adding a Document Queue to the Config.xml file

- To create a Document Queue in the Config.xml file locate an existing Document Queue entry e.g. Sales Document Queue.
- Copy all of the code between the <docsQueue> and </DocsQ> entries and paste immediately below the </DocsQ> code.
- In the DocsQ Description entry change the name of the queue as required.
  - <DocsQ Description="Shipment Document Queue"
- In the Location entry enter the Path to this queue as specified above.
  - Location="\\User specified queue location\Shipment Document Queue\">

#### Sample Code:

```

<docsQueue>
  <DocsQ Description="Shipment Document Queue" Location="\\User specified queue
location\Shipment Document Queue\">
    <EnablePDFTextExtractionToFile>>false</EnablePDFTextExtractionToFile>
    <BatchScanSplitType>Barcode</BatchScanSplitType>
    <BatchScanPageInterval>1</BatchScanPageInterval>
    <BatchScanBarcodePrefix></BatchScanBarcodePrefix>
    <BatchScanBarcodeDeleteSplitPage>>false</BatchScanBarcodeDeleteSplitPage>
    <FilenamePrefix></FilenamePrefix>

```

```
</DocsQ>
```

### 3.3.2 Enabling and Disabling Abbyy

To use OCR and barcode reading you need to have Abbyy installed and licensed on your system, this should have been done during the Zetadocs Server install, if you have not do so you should do so now. To turn Abbyy on and off simply change the EnableAbbyy entry, the options are true and false. Systems using OCR or barcodes need to ensure that Abbyy support is enabled.

#### Sample Code:

```
<EnableAbbyy>>true</EnableAbbyy>  
<EnableAbbyy>>false</EnableAbbyy>
```

For our Shipment Queue setup we would set this value to true so that document batches are scanned and barcodes processed.

### 3.3.3 Batch Scanning Settings

This setting is specific to each document queue. Each document queue can be configured to use only one type of splitting method.

#### Configuring the Batch Scan Split Type

The splitting methods are:

- Off - The default, the Zetadocs Server doesn't process any batches present in the Batch Split folder of your document queue.
- Barcode - The Zetadocs Server splits the batches on every page that contains a barcode.
- BarcodeValue - The Zetadocs Server splits the batches every time the barcode being read has a different value to the previous barcode read.
- PageInterval - Zetadocs Server splits the batches using an interval defined by the BatchScanPageInterval parameter, e.g. every 3 pages see below for details.

#### Sample Code:

```
<BatchScanSplitType>Off</BatchScanSplitType>  
<BatchScanSplitType>Barcode</BatchScanSplitType>  
<BatchScanSplitType>BarcodeValue</BatchScanSplitType>  
<BatchScanSplitType>PageInterval</BatchScanSplitType>
```

For our Shipment Queue setup we would set this value to BarcodeValue so that document batches would be split on each unique barcode.

#### Configuring the Batch Scan Page Interval

This value determines the page interval number to be used when processing a document using the PageInterval splitting method described above.

```
<BatchScanPageInterval>"Value"</BatchScanPageInterval>
```

Where “Value” is a positive number, note this field cannot be left blank. For those not using page interval splitting we recommend adding the value 1, this will be ignored unless the BatchScanSplitType has been set to PageInterval. For our Shipment Queue setup we would set this value to 1 .

### Enabling Automatic deletion of the split page

This setting is specific to each document queue. When using the Barcode and BarcodeValue splitting methods, if you use a splitting page containing a barcode, you may want to delete the split page. This can be set under the BatchScanBarcodeDeleteSplitPage parameter. This feature is often used when splitter pages have been added throughout a batch to define when each multipage document ends and prevents these pages from ending up in the archive.

**Note:** By activating this feature any page containing a barcode used to split the batch will be deleted, if you are using barcode splitting all pages with a readable barcode will be deleted after splitting. If using the barcode prefix option, only those pages with the correct barcode prefix will be used to split on and will then, if not a single page, be deleted.

```
<BatchScanBarcodeDeleteSplitPage>true</BatchScanBarcodeDeleteSplitPage>  
<BatchScanBarcodeDeleteSplitPage>false</BatchScanBarcodeDeleteSplitPage>
```

For our Shipment Queue setup we would set this value to False so that documents with barcodes on wouldn't be lost.

## 3.3.4 Barcode Settings

We will cover adding barcodes to your NAV or BC reports [later in this guide](#), if you are intending to use barcode splitting on captured documents to streamline your processes you need to configure the Zetadocs Server accordingly using the options below.

### Barcode Detection

Zetadocs can detect barcodes if the OCR module has been installed, this option determines whether the Zetadocs Server searches for readable barcodes or not.

```
<EnableBarcodeDetection>true<EnableBarcodeDetection>  
<EnableBarcodeDetection>false<EnableBarcodeDetection>
```

For our Shipment Queue setup we would set this value to true so that document batches would be split and processed on barcode detection.

### Barcode Prefixes

This setting is specific to each document queue. When using the Barcode and BarcodeValue splitting methods, a barcode prefix can be set to ensure the Zetadocs Server only processes barcodes that contain the prefix. The prefix is defined under the BatchScanBarcodePrefix parameter shown below.

```
<BatchScanBarcodePrefix>Prefix</BatchScanBarcodePrefix>
```

**Note:** The Prefix value is case sensitive.

For our Shipment Queue setup we would leave this value blank as we are not setting up a system with prefixes. Save the updated Config.xml file when finished.

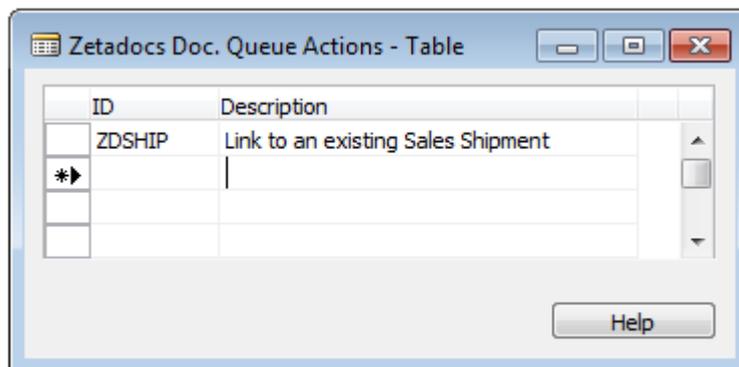
## 3.4 Custom Document Queues

To start processing documents from a queue we need to add the queue settings to Zetadocs. Let's continue our example by configuring a Shipment Queue in NAV or BC.

### Document Queue Actions

First you need to decide on the actions that you will implement for your queue. The action that is performed is up to you. Obviously the typical action is to associate (often referred to as linking) the document with a record in NAV or BC and then to archive the document. There is potential to use the document queues as events that will start other operations or modify NAV or BC data tables. It is entirely up to you what you want to implement in your actions. In this example we will implement just one linking action. To add new actions you need to run the Zetadocs Doc. Queue Actions table from the Object Designer and enter the item.

- Run table 9009991
- Add a new action ID ZDSHIP and its description



Creating new Document Queue Actions codes

### Setting up the Queue in NAV

Next you need to configure the Queue in NAV or BC.

- Open the Zetadocs Document Queue Card
  - For NAV 2013, NAV 2013 R2 or NAV 2015 select Administration > Application Setup > Zetadocs Setup > Zetadocs Document Queue List and select the queue you wish to modify to open the card.
  - For NAV 5 and 6 select Administration > Application Setup > Zetadocs Setup > Zetadocs Document Queue Setup.
- Create a new queue record noting the No. allocated.
- Enter the rest of the details as shown below. Note that the Caption added here will be displayed to the user when processing in the document queue and so it should give sufficient information to the user about the action. However, when there is only one action it is performed by default.

Caption	Menu Code ID
▶ Link to existing Sales Shipment	ZDSHIP

Document Queue Settings

### 3.4.1 Adding Document Queues to the Menusuite

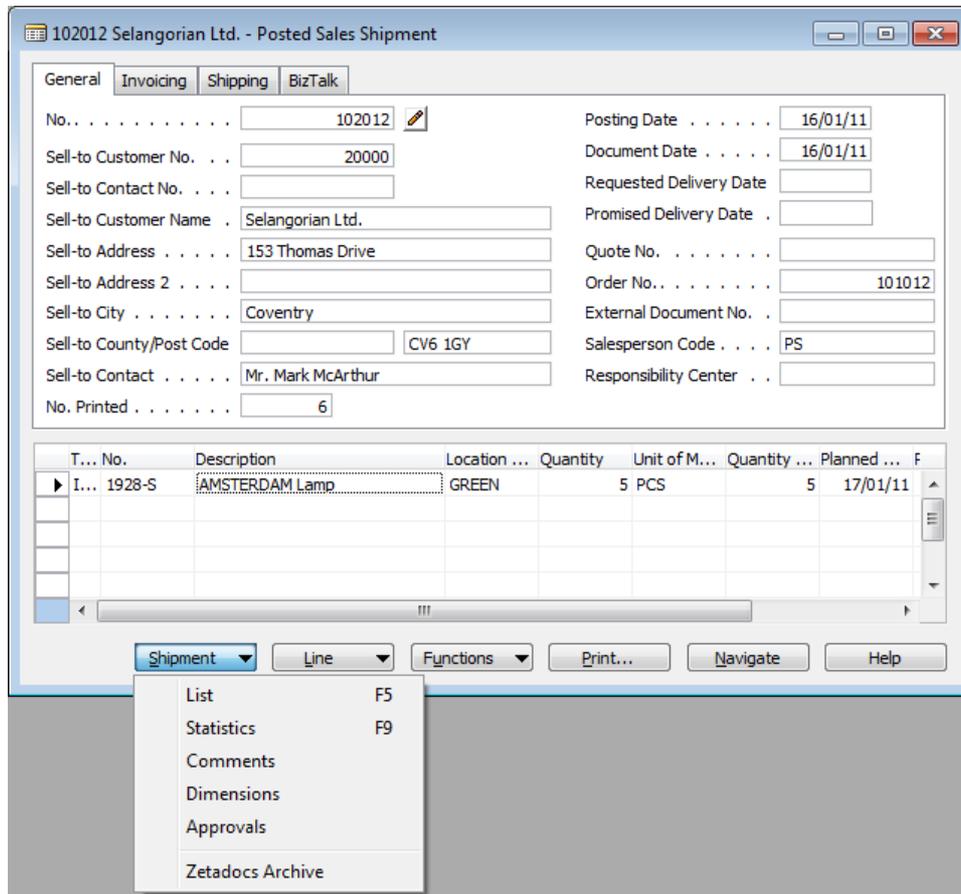
To enable users to quickly access your document queues you can add them to your menusuite, to do this simply follow the simple steps below.

- Open the object designer within NAV or BC and select the menusuite option.
- Create a new menusuite or edit an existing one as per your requirements.
- If creating a new one select the New button and then select the Design level appropriate to your install, e.g. Company.
- Locate the area in the Navigation pane where you wish to add the item to the menusuite.
- Right click on the area and select Create item.
- Select the object type of Form and pick the document queue in question from the object id field.
- Select OK when satisfied and the use ctrl+shift and the arrow keys to move your new item around the menusuite until you are happy with its location.
- Press ctrl+s to save your changes.
- Repeat for any remaining document queues.

Note: Document Queues will not be available via the menusuite until you have closed and restarted NAV or BC.

### 3.4.2 Archive Search and Retrieval

To allow a user to search, retrieve and view captured documents from a NAV or BC record form it is necessary to modify the form to provide a menu item that will do the search. For example the Posted Sales Shipment record form would be modified to add a Zetadocs Archive item on the Shipment menu, as below.



Posted Sales Shipment form showing Zetadocs Archive menu item

In our example scenario we need to modify the Posted Sales Shipment form (and page if using the Windows Client) to show the related document from the archive. If you are using the Windows Client, then make the equivalent changes in the page editor for the Windows Client record page.

### 3.4.3 Implementing Custom Document Queue Actions

#### Overview

In this section we write the code to enable document linking and archiving for a particular document queue action. Zetadocs Capture Essentials comes with some standard Sales and Purchase Order actions already implemented. This section shows you how to implement your own custom actions. There are a number of functions that Zetadocs calls in the Zetadocs-Capture Customize codeunit that you need to implement. The steps to follow are:

- Write code in the GetLink function

- Write code in the GetAdditionalMetadata function
- Add permissions to codeunit
- Test document archiving

The following sections provide more detail on each step.

### Write code in the GetLink function

For each action that is defined for the Archive button, you need to write some code in the function **GetLink**, to carry out that action in NAV or BC.

For example, if the action is to link the captured document to an existing record, then in this function you would code the lookup that displays a form to allow the user to select the record and then return that RecordID. Zetadocs will then use the record ID during the document archive and extract the correct metadata.

The **GetLink** function has the following parameters:

Name	Type	Sub-type	Read/Write	Description
ZdArchiveContext	Record	Zetadocs Archive Context	Read	A read only runtime table to provide context information like the queue the item is in and the menu item that was clicked by the user.
ZdLinkResult	Record	Zetadocs Link Result	Read/Write	A writable runtime table to allow you to return the un-typed RecordID reference to a NAV or BC record, error states and messaging.
ZdControlFile	Text [250]		Read	Path to the DQF file for the document queue item (XML file). Provided to allow you to fetch other information about converted files.

#### Sample Code:

This sample code shows an implementation of the GetLink function allowing the user to select the associated Sales Shipment record. Note that this function will be called for every Action. You will need to add a CASE statement using the Menu Code ID to identify the action the user has selected and write the appropriate code against each case.

```

PROCEDURE GetLink@100000007(ZdArchiveContext@100000000 : Record 9009992;VAR
ZdLinkResult@100000001 : Record 9009997;ZdControlFile@100000002 : Text[250]);
VAR
    SalesShipmentRec@100000003 : Record 110;
    RecRef@100000004 : RecordRef;
BEGIN

```

```

ZdUtilities.Log(0, 'C9009964 - Link');
// Menu Code ID find out what action to perform

// if there are any errors during processing set the error flag in ZdLinkResult to
TRUE
// and set the error description field to a useful error message

CASE ZdArchiveContext."Menu Code ID" OF
  'ZDSHIP': // link to existing sales shipment
    BEGIN
      // make sure the sales shipment record does not have any filters
      SalesShipmentRec.RESET;
      // run the lookup form and check the result
      IF FORM.RUNMODAL(0, SalesShipmentRec) = ACTION::LookupOK THEN
        BEGIN
          // if the user selected a record use a RecordRef to get the RecordID for the
record
          // and modify the ZdLinkResult record to include it
          RecRef.GETTABLE(SalesShipmentRec);
          ZdLinkResult."Record ID" := RecRef.RECORDID;
        END
      ELSE
        BEGIN
          // if the user cancelled, set the cancel flag in ZdLinkResult to true
          ZdLinkResult.Cancel := TRUE;
        END;
      // Call MODIFY to save the changes to the record
      ZdLinkResult.MODIFY;
    END;
  END;
END;

```

### Write code in the `GetAdditionalMetadata` function

**Please note:** this feature is not available when using the Zetadocs Archive.

Zetadocs calls the function **GetAdditionalMetadata** to get the values of metadata to be added to SharePoint when the document is archived.

The **GetAdditionalMetadata** function has the following parameters:

Name	Type	Sub-type	Read/Write	Description
ZdArchiveContext	Record	Zetadocs Archive Context	Read	A read only runtime table to provide context information like the queue the item is in and the menu item that was clicked by the user.
ZdLinkResult	Record	Zetadocs Link Result	Read/Write	A writable runtime table to allow you to return the un-typed RecordID reference to a NAV or BC record, error states and messaging.

ZdArchiveMetadata	Record	Zetadocs Archive Metadata	Read/Write	A writable runtime table to allow you to modify the field values being archived to SharePoint and to fill in custom metadata that has multi-level table relations.
-------------------	--------	---------------------------------	------------	--

### Sample Code:

This sample code shows an implementation of GetAdditionalMetadata that sets the values for the metadata for Record Type, Record Number, Company Number, Name and Organization. Note that this function will be called for every Action. You will need to add a CASE statement using the Record ID to identify the linked record and then write the appropriate code against each case to pull the specific metadata required.

```
PROCEDURE GetAdditionalMetadata@1000000005(ZdArchiveContext@1000000000 : Record
9009992;ZdLinkResult@1000000001 : Record 9009997;VAR ZdArchiveMetadata@1000000002 : Record
9009993);
```

```
VAR
```

```
  RecRef@1000000003 : RecordRef;
```

```
  fRef@1000000004 : FieldRef;
```

```
BEGIN
```

```
  ZdUtilities.Log(0, 'C9009964 - GetAdditionalMetadata');
```

```
  // Use the record id to get the table number. From the table number we know which
  fields to get the metadata from.
```

```
  // We are using short codes for the standard Zetadocs SharePoint fields.
```

```
  // These codes will be translated into the correct field names.
```

```
  // For custom fields just put the full field name in and it will be used.
```

```
IF RecRef.GET(ZdLinkResult."Record ID") THEN
```

```
BEGIN
```

```
  CASE RecRef.NUMBER OF
```

```
    110: // Sales Shipment Header Table
```

```
      BEGIN
```

```
        // First fill in the record type
```

```
        ZdArchiveMetadata.INIT;
```

```
        ZdArchiveMetadata.Name := 'ZetadocsRecordType';
```

```
        ZdArchiveMetadata.Value := 'Posted Sales Shipment';
```

```
        ZdArchiveMetadata.Type := ZdArchiveMetadata.Type::STRING;
```

```
        ZdArchiveMetadata.INSERT;
```

```
        // Field 3- "No."
```

```
        fRef := RecRef.FIELD(3);
```

```
        ZdArchiveMetadata.INIT;
```

```
        ZdArchiveMetadata.Name := 'RECORDNUMBER';
```

```
        ZdArchiveMetadata.Value := fRef.VALUE;
```

```
        ZdArchiveMetadata.Type := ZdArchiveMetadata.Type::NUMBER;
```

```
        ZdArchiveMetadata.INSERT;
```

```
        // Field 2 - "Sell-to Customer No."
```

```
        fRef := RecRef.FIELD(2);
```

```
        ZdArchiveMetadata.INIT;
```

```
        ZdArchiveMetadata.Name := 'COMPANYNUMBER';
```

```
        ZdArchiveMetadata.Value := fRef.VALUE;
```

```

ZdArchiveMetadata.Type := ZdArchiveMetadata.Type::STRING;
ZdArchiveMetadata.INSERT;

// Field 84 - "Sell-to Contact"
fRef := RecRef.FIELD(84);
ZdArchiveMetadata.INIT;
ZdArchiveMetadata.Name := 'NAME';
ZdArchiveMetadata.Value := fRef.VALUE;
ZdArchiveMetadata.Type := ZdArchiveMetadata.Type::STRING;
ZdArchiveMetadata.INSERT;

// Field 79 - "Sell-to Customer Name"
fRef := RecRef.FIELD(79);
ZdArchiveMetadata.INIT;
ZdArchiveMetadata.Name := 'ORGANIZATION';
ZdArchiveMetadata.Value := fRef.VALUE;
ZdArchiveMetadata.Type := ZdArchiveMetadata.Type::STRING;
ZdArchiveMetadata.INSERT;
    END;
END;
END;
END;

```

#### Add Permissions to codeunits

It is advisable to check that the correct permissions are granted to the Zetadocs-Capture Customize and the Zetadocs-Capture Integration codeunits for the table data of the records you wish to link to. This ensures that the codeunits will be able to read all the record information necessary as part of your custom link actions.

### 3.4.4 Testing Document Archiving

At this stage you should test that the functionality you have added works by doing the following:

- Scan a document to your new document queue.
- Check that the document is listed in the document queue within NAV or BC and can be viewed correctly.
- With the document selected click the Archive button and complete the actions.
- Browse to the archive and check that the document has been archived successfully in the correct location.
- View the document properties and check these are as expected.
- Check the Zetadocs Archive option from the record form finds the archived document.
- (Finally, when you have completed your testing delete any test documents from the archive.)

## 3.5 Automation Using Barcodes

Zetadocs supports the addition of barcodes to reports printed from NAV or BC using a barcode font to print unique information about an NAV or BC object. This places a barcode into the report output, which can be used in a number of manners. For example, with documents like invoices or sign off sheets which are distributed from NAV or BC before being signed or filled in and returned. In these cases when the document is added back into the system document queue via scanning or email etc, it is captured back into the system and OCR'd. This process detects the

barcode and decodes the information contained within it. It is then used to link the returned document to the original document in the archive. This enables users to track documents through a variety of business processes.

## Barcode Formats

Zetadocs for NAV or BC supports Code 128 format barcodes only, however the OCR software has the capacity to read and process those listed below. We recommend a minimum of Font-size 32 and a scanner resolution dpi of 300 or greater. It should also be noted that some barcode formats only support upper case letters, such as Code 39, if using such a barcode format please ensure you only use supported characters in your barcodes.

### Numeric-only barcodes

- EAN-13 (European Article Numbering, an international retail product code)
- EAN-8 (compressed version of EAN code for use on small products)
- UPC-A (Universal Product Code seen on almost all retail products in the USA and Canada)
- UPC-E (compressed version of UPC code for use on small products)
- Code 11 (used primarily for labelling telecommunications equipment)
- Interleaved 2 of 5 (compact numeric code, widely used in industry, air cargo, other applications)
- Industrial 2 of 5 (older code not in common use)
- Standard 2 of 5 (older code not in common use)
- Codabar (older code often used in library systems, sometimes in blood banks)
- Plessey (older code commonly used for retail shelf marking)
- MSI (variation of the Plessey code commonly used in USA)
- PostNet (used by U.S. Postal Service for automated mail sorting)

### Alphanumeric barcodes

- Code 39 (de facto standard for Government, Manufacturing, BarCode Industry, Education, and Business applications, in use world-wide)
- Code 93 (compact code similar to Code 39)
- Code 128 (very capable code, excellent density, high reliability; in very wide use world-wide)
- LOGMARS (same as Code 39, this is the U.S. Government specification)

## Barcode Options

- The Zetadocs server searches for barcode information when processing graphics and PDF image files. Any barcodes which are included in the file are saved alongside the document, and can be used by custom processing functions to identify the document and automate its processing.
- A typical usage is to encode the NAV or BC document number directly – eg adding the shipping note number as a barcode. However for more complex requirements a separate ID could be stored instead – eg storing a lookup key into a custom table within NAV or BC which is used to store custom information about the document. Barcode information is passed transparently to the custom code when processing for maximum flexibility.
- Zetadocs is supplied with a standard barcode font. When used with this font, the barcode recognition supports the following features:
  - Encoding of variable length data strings, including letters A-Z (from ANSI standard Western European character set), digits 0-9, plus a selection of punctuation characters.
  - Barcode including text (user readable) version for manual checking or processing.

- Protection against misreading, using automatically generated checksum data.
- Support for documents scanned upside down.
- Most other common barcode formats can also be supported. Please contact Equisys for further details if there is a requirement to use a specific barcode format.
- Barcode processing requires the OCR add-on module.
- Use of the barcode to identify and process the document requires customization work by the VAR using the Zetadocs Capture SDK, as described below.

**Note:** This method works for most reports; however, due to the customizable nature of reports it may not fit all reports exactly. Therefore, the location of where to insert the code below may need to vary but the code will remain the same.

### 3.5.1 Installing the Barcode Font

Barcodes can be added to reports by simply using a suitable font. The chosen barcode font must be installed on each client PC that will be used to print the report. There are many different types of barcode font available, but we have found that Code 39 barcodes work well with Zetadocs. You can download and install a free Code 39 barcode font from <http://www.barcodesinc.com/free-barcode-font/>. Choose either the standard or extended font and follow the included instructions to install the font on each client PC. Barcode fonts often have limited character support e.g. A-Z and 0-9. So please ensure that the characters you wish to use are supported by the font.

### 3.5.2 Modify an NAV or BC Report to Add a Barcode

Before you modify your report, you need to decide what unique piece of information you will use for the barcode, so that the printed report can subsequently be identified in the capture stage. This will usually be the record number of the NAV or BC record. For example for the Shipment Note report you could use the Shipment record number from the Posted Sales Shipment Record.

Follow these instructions to modify a report to add a barcode:

- Open the Development Environment/classic client and open the Object Designer (Shift+F12).
- Locate the report you wish to modify in the Object Designer and select Design.
- In the Report Designer select View > Sections.
- Open the toolbox View > Toolbox.
- Select the text box tool and create one in a suitable location for the barcode - we suggest the main header section.
- Right-click on the text box and select properties.
- Set the text box FontName to the barcode font name  
e.g. FontName = Free 3 of 9 Extended
- Edit the FontSize to a suitable value, we recommend a minimum size of 32.
- Set the text box SourceExpr property to the value to be used for the barcode  
e.g. SourceExpr = STRSUBSTNO(\*SHIP%1\*,"Sales Shipment Header"."No.")
- Ensure that the barcode starts with and ends with “\*”, otherwise the barcode will not be recognized by the scanner.

- We also suggest that you add another text box below the barcode to display the value in a standard font so that it is also human readable.
- Save the changes to the report.
- Test that the report prints OK and that the barcode is clearly legible.

If you are using the Windows Client, then make the equivalent changes in the report editor for the Windows Client report.

### 3.5.3 Adding Code to the GetLinkDisplayString Function

Zetadocs calls the function **GetLinkDisplayString** to allow you to return a suitable text string to be displayed in the document queue. If no value is returned by this function, then the display will default to the barcode text for the first barcode found in the captured document. If the document contains multiple barcodes, these will be represented as multiple records in the table.

The **GetLinkDisplayString** function has the following parameters:

Parameter	Type	Sub-type	Read/Write	Description
ZdDocQueueBarcodes	Record	Zetadocs Doc. Queue Barcodes	Read	A read only runtime table that allows you to access the values of all barcodes from the document queue item.
Return Value	Text[120 ]		Write	The text to display in the document queue "Link" column.

- Open the Development Environment/classic client and open the Object Designer (Shift+F12).
- Locate the Codeunit ID:9009964 Zetadocs-Capture Customise in the Object Designer and select Design.
- Scroll down to the GetLinkDisplayString section.
- You will notice the code matches that displayed below, this indicates that it is setup for the Shipment Report, which we provide as a sample report.
- To add support for other reports we need to add additional if statements to ensure your system is configured to deal with barcodes which start with other details.

#### Sample Code:

This sample code shows an implementation of GetLinkDisplayString that returns a display string for a Shipment record.

```
// Call reset to clear any filters that may be on the barcodes temporary table
ZdDocQueueItemBarcodes.RESET;

IF NOT ZdDocQueueItemBarcodes.FIND('-') THEN
BEGIN
    // There are no barcodes so we leave the link string blank
    EXIT('');
END;
```

```

info := ZdDocQueueItemBarcodes.Value;

// if the barcode starts with SHIP
IF (STRLEN(info) > 4) AND (COPYSTR(info, 1, 4) = 'SHIP') THEN
BEGIN
    info := STRSUBSTNO(strSalesShipment, COPYSTR(info, 5));
END;
END;

```

### Customising the GetLinkDisplayString function

The document specific section of code below contains an IF statement which enables the codeunit to add additional information to the document queue when it detects a barcode which begins with SHIP. The makeup of this string is obviously determined by the unique value you wish to use for your barcode.

#### String in the report which generates the barcode

```
SourceExpr = STRSUBSTNO('*SHIP%1*', "Sales Shipment Header"."No.")
```

This would for example generate a value of \*SHIP 102028\*.

#### Corresponding code required in the codeunit

```

// if the barcode starts with SHIP
IF (STRLEN(info) > 4) AND (COPYSTR(info, 1, 4) = 'SHIP') THEN
BEGIN
    info := STRSUBSTNO(strSalesShipment, COPYSTR(info, 5));
END;

```

The code then detects that our barcode starts with SHIP and acts accordingly.

### Adding additional IF statements for other reports

If then you wanted to add barcode support to another report then you would need to modify the report as per the steps in [section 3.5.2](#) adding the SourceExpr value shown below. Note you can use whatever identifier seems appropriate in this case we have chosen to setup a Sales Invoice report so have chosen SINV and have replaced Sales Shipment Header for Sales Invoice Header.

```
SourceExpr = STRSUBSTNO('*SINV%1*', "Sales Invoice Header"."No.")
```

We then need to add an additional if statement to match this:

```

// if the barcode starts with SINV
IF (STRLEN(info) > 4) AND (COPYSTR(info, 1, 4) = 'SINV') THEN
BEGIN
    info := STRSUBSTNO(strSalesInvoice, COPYSTR(info, 5));
END;

```

This can be repeated as desired for any remaining reports.

### 3.5.4 Write Code in the GetAutoLink Function

Zetadocs calls the function **GetAutoLink** for each item in the queue so that you can determine if it can be automatically associated with a NAV or BC record. If you plan to have this function called automatically from the NAS server or from other codeunits, it is recommended that you do not include user input operations. For further information follow the reference in [section 3.9](#).

The **GetAutoLink** function has the following parameters:

Parameter	Type	Sub-type	Read/Write	Description
ZdArchiveContext	Record	Zetadocs Archive Context	Read	A read only runtime table to provide context information like the queue the item is in and the menu item that was clicked by the user.
ZdDocQueueBarcodes	Record	Zetadocs Doc. Queue Barcodes	Read	A read only runtime table that allows you to access the values of all barcodes from the document queue item.
ZdLinkResult	Record	Zetadocs Link Result	Read/Write	A writable runtime table to allow you to return the un-typed RecordID reference to a NAV or BC record, error states and messaging.
ZdControlFile	Text [250]		Read	Path to the DQF file for the document queue item (XML file). Provided to allow you to fetch other information about converted files.

#### Sample Code:

This sample code shows an implementation of GetAutoLink that finds and returns the matching record ID.

```
PROCEDURE GetAutoLink@1000000006(ZdArchiveContext@1000000002 : Record 9009992;VAR
ZdLinkResult@1000000001 : Record 9009997;ZdControlFile@1000000000 : Text[250];VAR
ZdDocQueueBarcodes@1000000003 : Record 9009995);
VAR
    msgNoBarcodes@1000000005 : TextConst 'DAN=Auto linking failed. No barcodes were
detected on the document.;DEU=Auto linking failed. No barcodes were detected on the
document.;ENU=Auto linking failed. No barcodes were detected on the document.;ESP=Auto
linking failed. No barcodes were detected on the document.;FRA=Auto linking failed. No
barcodes were detected on the document.;ITA=Auto linking failed. No barcodes were
detected on the document.;NLD=Auto linking failed. No barcodes were detected on the
document.;ENG=Auto linking failed. No barcodes were detected on the document.';
    msgCouldNotFindRecord@1000000006 : TextConst 'DAN=Auto linking failed. Could not
find record matching barcode: %1;DEU=Auto linking failed. Could not find record matching
```

```
barcode: %1;ENU=Auto linking failed. Could not find record matching barcode: %1;ESP=Auto
linking failed. Could not find record matching barcode: %1;FRA=Auto linking failed.
Could not find record matching barcode: %1;ITA=Auto linking failed. Could not find record
matching barcode: %1;NLD=Auto linking failed. Could not find record matching barcode: %
1;ENG=Auto linking failed. Could not find record matching barcode: %1';
barcodeVal@1000000004 : Text[80];
ShipmentRec@1000000007 : Record 110;
RecRef@1000000008 : RecordRef;
BEGIN
ZdUtilities.Log(0, 'C9009964 - AutoLink');
```

This code should be the same for every report and doesn't need repeating with each if statement.

```
// Use the barcodes that are passed in to find the record to link to.

// Call reset to clear any filters that may be on the barcodes temporary table
ZdDocQueueBarcodes.RESET;

IF NOT ZdDocQueueBarcodes.FIND('-') THEN
BEGIN
// There are no barcodes so don't modify the record.
// Processing will fall back to calling the Link function for the manual method.
EXIT;
END;

barcodeVal := ZdDocQueueBarcodes.Value;
```

### IF Statement

```
// If the barcode starts with SHIP it means it is a shipment note.
// The barcode will be in the format SHIP000000, where 000000 is the shipment number
IF (STRLEN(barcodeVal) > 4) AND (COPYSTR(barcodeVal, 1, 4) = 'SHIP') THEN
BEGIN
ShipmentRec.RESET;
ShipmentRec."No." := COPYSTR(barcodeVal, 5);
// If we can find the record that the barcode refers to
IF ShipmentRec.FIND THEN
BEGIN
// if the user selected a record use a RecordRef to get the RecordID for the
record
// and modify the ZdLinkResult record to include it
RecRef.GETTABLE(ShipmentRec);
ZdLinkResult."Record ID" := RecRef.RECORDID;
ZdLinkResult.MODIFY;
END;
END;

EXIT;
END;
```

### Modified IF Statement for the Sales Invoice

```
// If the barcode starts with SINV it means it is a Sales Invoice.
// The barcode will be in the format SINV000000, where 000000 is the Sales Invoice
number
IF (STRLEN(barcodeVal) > 4) AND (COPYSTR(barcodeVal, 1, 4) = 'SINV') THEN
```

```

BEGIN
  ShipmentRec.RESET;
  ShipmentRec."No." := COPYSTR(barcodeVal, 5);
  // If we can find the record that the barcode refers to
  IF ShipmentRec.FIND THEN
    BEGIN
      // if the user selected a record use a RecordRef to get the RecordID for the
record
      // and modify the ZdLinkResult record to include it
      RecRef.GETTABLE(ShipmentRec);
      ZdLinkResult."Record ID" := RecRef.RECORDID;
      ZdLinkResult.MODIFY;
    END;
  END;

  EXIT;
END;

```

### 3.5.5 Testing Document Auto Linking

At this stage you should test that the functionality you have added works by doing the following:

- Print a modified report that includes a barcode.
- Scan the printed report into your new document queue.
- Check that the document is listed in the document queue within NAV or BC and can be viewed correctly.
- Check that the scanned document displays the appropriate text in the Link column.
- Click the Archive button and complete the actions.
- Browse to the archive and check that the document has been archived successfully in the correct location.
- View the document properties and check these are as expected.
- (Finally, when you have completed your testing delete any test documents from the archive.)

## 3.6 Automation Using Extracted Text

Now that we have covered how you can use barcodes to automate some of your processes, we are going to demonstrate how text extraction on document capture can be used to add yet further automation. This section will deal with how to extract the text from a captured document and then an example of how this can be used to automate processes.

### Enabling Zetadocs Document Capture Text Extraction

Zetadocs Capture Plus supports the parsing of document text, this can be used for identification numbers and metadata to facilitate document recognition. This then allows the automated linking and archiving of documents with NAV or BC records. Implementing automated document recognition has many benefits including the streamlining of processes, by the automation of tedious repetitive activities and by automatically initiating internal workflows. To access this functionality you must enable text extraction on Zetadocs Document Queues configurations in the Zetadocs Server and then use a Zetadocs SDK function to get the path to the extracted text file.

### Enabling text extraction in the Zetadocs Server

Each configured queue in the Zetadocs Server can have text extraction enabled independently. This allows you to enable text extraction processing on the queues in which you expect to implement document recognition. Furthermore it means that the unnecessary processing of documents is avoided. It is therefore recommended that you create separate queues for documents for which you intend to implement document recognition and for those you do not.

### Configuring the Zetadocs Server

Zetadocs Server settings are modified using a Config.xml file, this is located in the Zetadocs Server installation location usually C:\Program Files (x86)\Zetadocs Server\Document Converter. Open this file in a text editing program such as Notepad.exe and locate the queue you wish to modify. To enable text extraction set the value of the xml element to **true** to enable text extraction. NOTE: Due to xml serialization the value is case sensitive and must be set to **true** or **false**. True or TRUE or any other variation will be invalid.

```
<EnablePDFTextExtractionToFile>>false</EnablePDFTextExtractionToFile>
```

### Accessing the extracted text file from NAV or BC

Once you have enabled the text extraction in the Zetadocs Server you will need to implement parsing of the text file in NAV or BC. This is done using the Zetadocs Capture Customize codeunit.

In the Zetadocs- Capture Customize codeunit trigger GetAutoLink() the path to the control file for queue items is available as the ZdControlFile :Text[250] parameter. Using this you can call a Zetadocs Utilities function to return the associated extracted text file path for that item.

```
textFilePath := ZdUtilities.GetExtractedTextFilePath(ZdControlFile);
```

You can now implement your own functionality to parse the text file for expected text metadata which will help you to identify the document and its associated record in much the same manner as is described for the barcode recognition feature of Zetadocs. For more information, please see [section 3.5](#).

## 3.6.1 Approvals in NAV or BC using Extracted Text

Now that we have successfully identified and extracted the information we require we can put it to work in NAV or BC, in this example we will be setting NAV or BC so that it starts an approval process.

In this example we will be looking at the following business process:

- Raise and send out a purchase order from NAV or BC using Zetadocs
- Received by the vendor who will raise a sales order and send the goods.
- They will attach a shipment note to the delivery and also send a purchase order.
- These documents will be Captured into the relevant Zetadocs Document Queue either via scanning of hardcopies or processing of other documents by the Zetadocs Server.
- When the Purchase Invoice is Captured, Zetadocs extracts the text including the original purchase order number.

- When the Purchase Invoice is processed and added to the Zetadocs Archive this set in motion the approval process in NAV or BC.

The existing Document Queues as setup by default do not support the initialization of the approval process. As such we need to adjust the Zetadocs-Capture Customize codeunit so that it begins the approval process when an invoice is archived.

### Creating the IsValueInFile function

- Select the Zetadocs-Capture Customize codeunit in the object designer, and select Design.
- Select View > C/AL Globals and choose the functions tab.
- Add the function IsValueInFile, details below.

Name	<b>IsValueInFile ( Value, FilePath )</b>
Description	Searches if a given <b>Value</b> is present in the file pointed by <b>FilePath</b>
Arguments	<b>Value</b> text[30]            The value to search for <b>FilePath</b> text[1024]        The associated text file to look in
Returned value	<b>True</b> if the number is present <b>False</b> if not

### Adding Code into the IsValueInFile function

Now that we have added the function into the codeunit we need to add the following code, open the C/AL editor (F9) and paste in the code below into the IsValueInFile section. We recommend that you review this now to gain an understanding of how it operates, comments have been included throughout to explain the various sections.

#### CODE:

```
IsValueInFile(text[30] Value; text[1024] FilePath)
{
  // Set file to be opened in text mode and for reading only
  file.TEXTMODE(TRUE);
  file.WRITEMODE(FALSE);

  // If we can't open the file, just return false as we cannot match a value
  IF NOT file.OPEN(FilePath) THEN
  BEGIN
    EXIT(FALSE);
  END;

  // Cycle through all lines in the file until we find te value we were
  // looking for or
  // we reach the end of the file
  REPEAT
    // Read each line at a time
    IF file.READ(line) <> 0 THEN
    BEGIN
      // Check if value appears anywhere in the line
      // NOTE: If values wrap over more than one line they will not be matched
      pos := STRPOS(line, Value);
      IF (pos <> 0) THEN
```

```

BEGIN
  // We've found the value we were looking for, close the file and
  // return true
  file.CLOSE;
  EXIT(TRUE);
END;
UNTIL file.POS = file.LEN; // Check if we have reached the end of the file

// We've reached the end of the file without finding the value we were
//looking for
// Close the file and return false
file.CLOSE;
EXIT(FALSE);
}

```

### 3.6.2 Creating the GetMatchingPurchaseOrder function

- Next return to the functions tab, View > C/AL Globals.
- Add the function GetMatchingPurchaseOrder, details below.

Name	<b>GetMatchingPurchaseOrder (Orders, ZdControlFile)</b>
Description	Searches through the ZdControlFile to find a matching Purchase Order
Arguments	<b>Orders</b> record List of opened purchase orders <b>ZdControlFile</b> text[1024] The control file of the document queue item
Returned value	<b>True</b> if the Order matches the DQF file <b>False</b> if not

#### Adding Code into the GetMatchingPurchaseOrder function

Now that we have added the function into the codeunit we need to add the following code, open the C/AL editor (F9) and paste in the code below into the GetMatchingPurchaseOrder section. We have included comments throughout to explain the various sections.

#### CODE:

```

GetMatchingPurchaseOrder (Record Orders, Text[1024] ZdControlFile)
{
  // Clear any filters on the Orders record
  Orders.RESET;

  // Set range so we are only looking at orders
  Orders.SETRANGE(Orders."Document Type",Orders."Document Type"::Order);

  // Make sure there are records to search through
  IF Orders.FIND('-') THEN
  BEGIN
    //Loop through records until we find a matching one or reach the end
  REPEAT
    // Check if the order number can be found in the file
    // NOTE: For this example we have changed the orders number

```

```

    //series so order numbers will look like:
    // "PO-00001" This should be unique in the file so we don't get
    //false matches
    IF IsValueInFile(Orders."No.", ZdUtilities.GetExtractedTextFilePath( ZdControlFile))
THEN
    BEGIN
    // We've found the right purchase order so return true
    EXIT(TRUE);
    END;
UNTIL Orders.NEXT = 0;
    END;

    // Reached the end of the orders without matching one, so return false
    EXIT(FALSE);
}

```

### 3.6.3 Adjusting the GetAutoLink Function

Now we have created and added code to our variables we need to adjust the **GetAutoLink** function. This is so it can look through all the Orders and find out if one of them matches up the Purchase Order number that is present in the associated file being archived. Scroll to the GetAutoLink section and add the code so that it surrounds the existing code as shown below.

#### CODE:

```

GetAutoLink (...)
{
    ZdUtilities.Log(0, 'C9009964 - AutoLink');

    CASE ZdArchiveContext."Zetadocs Doc. Queue No." OF
        'ZDQ0020', 'ZDQ0021':
            // Purchase Invoice and Purchase Receipt Document Queues
            BEGIN
                // Use the information in the control file to find a matching purchase order
                IF GetMatchingPurchaseOrder(Orders, ZdControlFile) THEN
                    BEGIN
                        // use a RecordRef to get the record id for the matching record
                        RecRef.GETTABLE(Orders);
                        ZdLinkResult."Record ID" := RecRef.RECORDID;
                        ZdLinkResult.MODIFY;
                    END;
                END
            ELSE
                BEGIN
                    ...
                    << EXISTING CODE HERE >>
                    ...
                END
            END;
    EXIT;
}

```

## 3.6.4 Adjusting the PostArchive Function

Next we need to add some extra code into the **PostArchive** function to send the Purchase Approval for a given item. Again this code needs to be pasted around the existing code.

```
PostArchive (...)  
{  
  // Check if the archive was successful  
  IF NOT (    ZdArchiveResult."Archive Document" AND ZdArchiveResult."Archive Succeeded")  
  THEN  
  BEGIN  
    EXIT;  
  END;  
  
  // Check this is the purchase invoice document queue  
  IF ZdArchiveContext."Zetadocs Doc. Queue No." = 'ZDQ0020' THEN  
  BEGIN  
    // Use a RecordRef to get values from the record  
    IF NOT RecRef.GET(ZdArchiveResult."Record ID") THEN  
    BEGIN  
      // Could not get the record from the record id  
      EXIT;  
    END;  
  
    // Get a fieldref to the document type field  
    fRef := RecRef.FIELD(1);  
  
    // Store the value as an integer so that we can compare it  
    DocType := fRef.VALUE;  
  
    // Check we have linked to an order  
    IF (DocType <> PurchaseOrder."Document Type"::Order) THEN  
    BEGIN  
      // If it is not an order we cannot start approval, so exit  
      EXIT;  
    END;  
  
    // Get a fieldref to the No. field  
    fRef := RecRef.FIELD(3);  
  
    // Store the order no  
    OrderNo := fRef.VALUE;  
  
    // Set range of the purchase order record so only looking at  
    // orders  
    PurchaseOrder.SETRANGE("Document Type", PurchaseOrder."Document Type"::Order);  
  
    // Set range so only looking at this specific order  
    PurchaseOrder.SETRANGE("No.", OrderNo);  
  
    // Check that an order exists in the range  
    IF PurchaseOrder.FIND('-') THEN  
    BEGIN  
      // Call into approvals management codeunit to start the approval process  
      IF ApprovalsMgmt.SendPurchaseApprovalRequest(PurchaseOrder) THEN;
```

```

    END;
END;

{
  ...
  << EXISTING CODE HERE >>
  ...
}

```

This should complete the setup required for the Purchase Invoices to generate an approval process, obviously the above example can be modified using different queue id's to effect different queues and thus document types.

## 3.7 Further Capture Customizations

### Overview

In this section a number of advanced functions are described that will help you to do further customisation of the document queue capture process. The functions available are:

- PreArchive
- PostArchive

### PreArchive

The **PreArchive** function is similar to the **OverrideSendResult** function in the delivery parts of the Zetadocs SDK. This function gives a writable copy of the final information to be used in the archiving of the document from the queue. It is at this point that you get the opportunity to override the archive location and content type.

There is an example of how to change the subfoldering plan based on document type and customer/vendor name. A subfolder string should be of the format 'sub/folder/path'. The Example shows how to set different paths depending on the type of capture. Please see the body of the function if you require further information.

The **PreArchive** function has the following parameters:

Parameter	Type	Sub-type	Read/Write	Description
ZdArchiveContext	Record	Zetadocs Archive Context	Read	A read only runtime table to provide context information like the queue the item is in and the menu item that was clicked by the user.
ZdArchiveSettings	Record	Zetadocs Archive Settings	Read/Write	A writable runtime table to allow you to modify properties of the archive action such as whether to archive and the archive location.

## PostArchive

Zetadocs calls the **PostArchive** function to inform you of the archiving result and to allow you to implement workflow based on that information.

The **PostArchive** function has the following parameters:

Parameter	Type	Sub-type	Read/Write	Description
ZdArchiveContext	Record	Zetadocs Archive Context	Read	A read only runtime table to provides context information like the queue the item is in and the menu item that was clicked by the user.
ZdArchiveResult	Record	Zetadocs Archive Result	Read/Write	A writable runtime table to allow you to find out if the archive was successful and get the URL of the archived file.

## 3.8 Capture Reference

### Overview

This section gives details of the records used by the capture functions:

- Zetadocs Archive Context
- Zetadocs Archive Metadata
- Zetadocs Archive Result
- Zetadocs Archive Settings
- Zetadocs Doc. Q. Item Barcode
- Zetadocs Link Result

### Zetadocs Archive Context

E.. Field No.	Field Name	Data Type	Length	Description
1	No.	Integer		Always 1 as table only used to ...
10	Zetadocs Doc. Queue No.	Code	10	Doc Queue that the click is from
11	Menu Code ID	Code	10	Menu Item that was clicked
40	UIEnabled	Boolean		Whether or not UI should be all...
50	CaptureType	Option		Type of Capture Factbox or Do...

Zetadocs Archive Context Table

The Zetadocs Archive Context table has the following fields:

- No.: Table record entry - please ignore
- Zetadocs Doc. Queue No. : Foreign Key to Zetadocs Document Queue table to allow you to know from which document queue the Link function has been called.
- Menu Code ID: Foreign Key to Zetadocs Document Queue Menu Item No. table to allow you to know which menu item had been selected by the user. Blank if used with GetAutoLink function.
- UIEnabled: Indicates whether UI (User Interface) dialogs should be shown or suppressed.
- CaptureType: Option value. It is the type of Capture Factbox or Document Queue.

## Zetadocs Archive Metadata

**Please note:** this feature is not available when using the Zetadocs Archive.

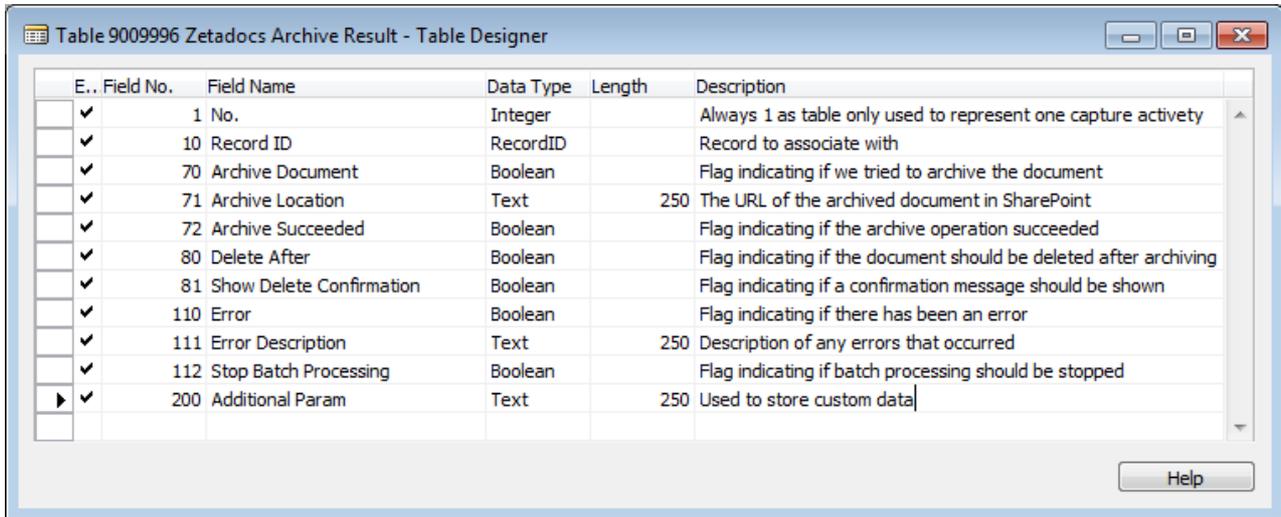
E.. Field No.	Field Name	Data Type	Length	Description
1	Name	Text	100	
2	Value	Text	250	
3	Type	Option		

Zetadocs Archive Metadata Table

The Zetadocs Archive Metadata table has the following fields:

- Name: Archive column name in SharePoint
- Value: The metadata value expressed as a string
- Type: The metadata type. Available types are NUMBER, DATE and STRING.

## Zetadocs Archive Result



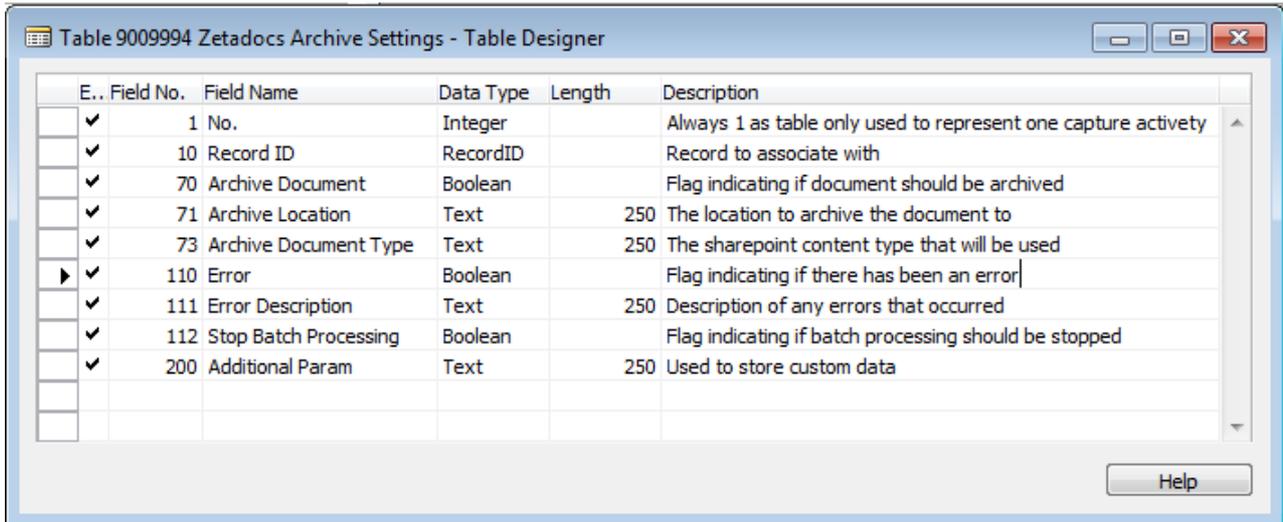
E.	Field No.	Field Name	Data Type	Length	Description
✓	1	No.	Integer		Always 1 as table only used to represent one capture activity
✓	10	Record ID	RecordID		Record to associate with
✓	70	Archive Document	Boolean		Flag indicating if we tried to archive the document
✓	71	Archive Location	Text	250	The URL of the archived document in SharePoint
✓	72	Archive Succeeded	Boolean		Flag indicating if the archive operation succeeded
✓	80	Delete After	Boolean		Flag indicating if the document should be deleted after archiving
✓	81	Show Delete Confirmation	Boolean		Flag indicating if a confirmation message should be shown
✓	110	Error	Boolean		Flag indicating if there has been an error
✓	111	Error Description	Text	250	Description of any errors that occurred
✓	112	Stop Batch Processing	Boolean		Flag indicating if batch processing should be stopped
▶	200	Additional Param	Text	250	Used to store custom data

Zetadocs Archive Result Table

The Zetadocs Archive Result table has the following fields:

- No.: Table record entry - please ignore
- Record ID: Un-typed record ID of the NAV or BC record that the captured document is associated with.
- Archive Document: Indicates if the captured document is to be archived.
- Archive Location: The URL of the archived document. **Please note:** this feature is not available when using the Zetadocs Archive.
- Archive Succeeded: Indicates if the archive was successful.
- Delete After: Indicates if the document should be deleted
- Show Delete Confirmation: Indicates to Zetadocs whether prompt the user for confirmation of delete.
- Error: Indicates to Zetadocs whether an error occurred during the function.
- Error Description: A description of the error that occurred.
- Stop Batch Processing: Used only when auto linking where this Boolean indicates whether batch processing should stop or continue.
- Additional Param: Additional parameter to allow extra data to be passed between Zetadocs functions.

## Zetadocs Archive Settings



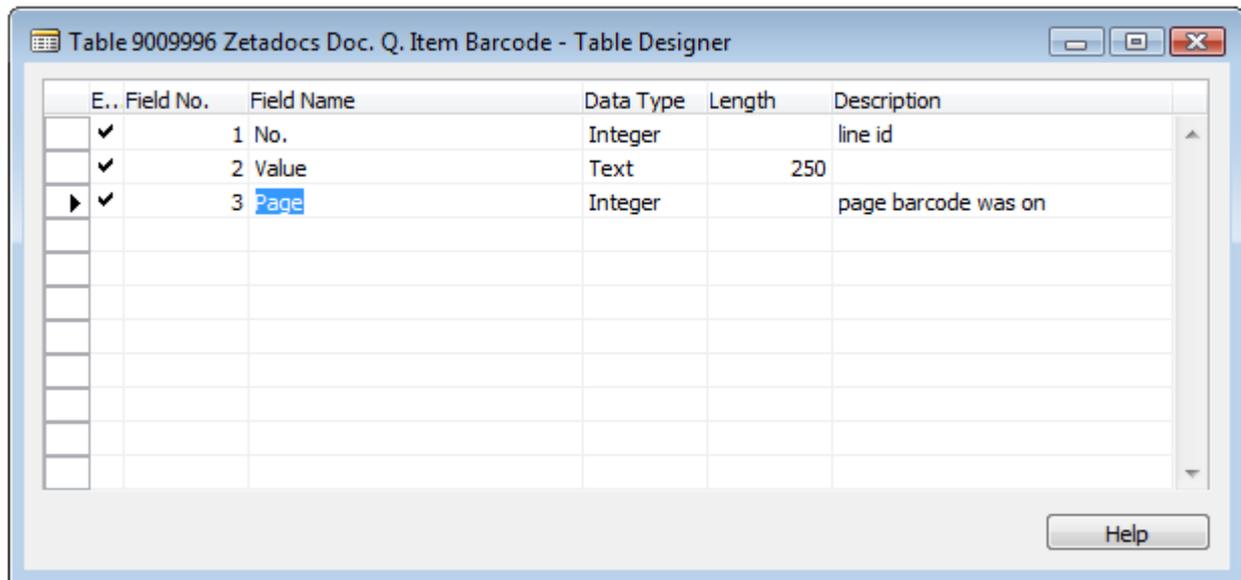
E.	Field No.	Field Name	Data Type	Length	Description
✓	1	No.	Integer		Always 1 as table only used to represent one capture activity
✓	10	Record ID	RecordID		Record to associate with
✓	70	Archive Document	Boolean		Flag indicating if document should be archived
✓	71	Archive Location	Text	250	The location to archive the document to
✓	73	Archive Document Type	Text	250	The sharepoint content type that will be used
▶	110	Error	Boolean		Flag indicating if there has been an error
✓	111	Error Description	Text	250	Description of any errors that occurred
✓	112	Stop Batch Processing	Boolean		Flag indicating if batch processing should be stopped
✓	200	Additional Param	Text	250	Used to store custom data

**Zetadocs Archive Settings Table**

The Zetadocs Archive Settings table has the following fields:

- No.: Table record entry - please ignore
- Record ID: Un-typed record ID of the NAV or BC record that the captured document is associated with.
- Archive Document: Indicates if the captured document is to be archived.
- Archive Location: The location to archive the document to.
- Archive Document Type: The SharePoint content type to be used when archiving.
- Error: Indicates to Zetadocs whether an error occurred during the function.
- Error Description: A description of the error that occurred.
- Stop Batch Processing: Used only when auto linking where this Boolean indicates whether batch processing should stop or continue.
- Additional Param: Additional parameter to allow extra data to be passed between Zetadocs functions.

## Zetadocs Doc. Q. Item Barcode



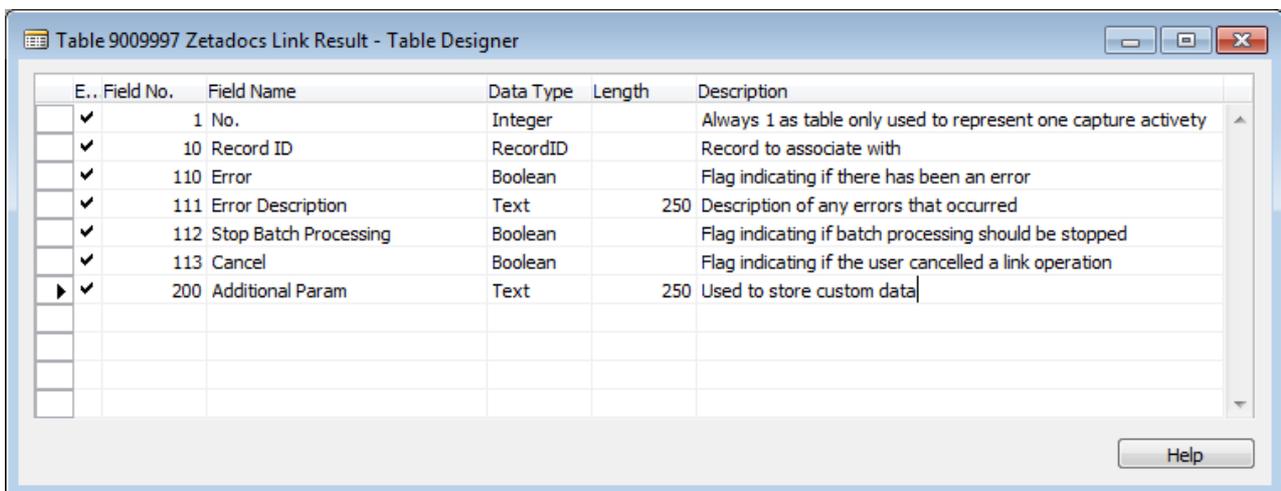
E..	Field No.	Field Name	Data Type	Length	Description
<input checked="" type="checkbox"/>	1	No.	Integer		line id
<input checked="" type="checkbox"/>	2	Value	Text	250	
<input checked="" type="checkbox"/>	3	Page	Integer		page barcode was on
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					

Zetadocs Doc. Q. Item Barcode Table

The Zetadocs Doc. Q. Item Barcode table has the following fields:

- No.: Table record entry - please ignore
- Value: The barcode expressed as a string
- Page: The page number the barcode was on

### Zetadocs Link Result



E..	Field No.	Field Name	Data Type	Length	Description
<input checked="" type="checkbox"/>	1	No.	Integer		Always 1 as table only used to represent one capture activity
<input checked="" type="checkbox"/>	10	Record ID	RecordID		Record to associate with
<input checked="" type="checkbox"/>	110	Error	Boolean		Flag indicating if there has been an error
<input checked="" type="checkbox"/>	111	Error Description	Text	250	Description of any errors that occurred
<input checked="" type="checkbox"/>	112	Stop Batch Processing	Boolean		Flag indicating if batch processing should be stopped
<input checked="" type="checkbox"/>	113	Cancel	Boolean		Flag indicating if the user cancelled a link operation
<input checked="" type="checkbox"/>	200	Additional Param	Text	250	Used to store custom data
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					

Zetadocs Link Result Table

The Zetadocs Link Result table has the following fields:

- No.: Table record entry - please ignore
- Record ID: Un-typed record ID of the NAV or BC record that the captured document is associated with.
- Error: Indicates to Zetadocs whether an error occurred during linking.
- Error Description: A description of the error that occurred.
- Stop Batch Processing: Used only when auto linking where this Boolean indicates whether batch processing should stop or continue.

- Cancel: Indicates whether the Archive action should be cancelled.
- Additional Param: Additional parameter to allow extra data to be passed between Zetadocs functions.

## 3.9 Capture Automation

### Summary

You can automatically archive linked items from a Zetadocs Document Queue programmatically.

The example provided in the following steps describes how to create a NAV or BC Codeunit that contains a function that calls the Zetadocs Capture Interface Codeunit. You can extend or modify the example to suit your needs or requirements.

### Steps

The steps described in the following section assume that Zetadocs is correctly setup and configured to use Zetadocs Capture Plus. They have been written for a NAV 2015 system configured to use Zetadocs Capture. Similar steps may apply to later versions of NAV or BC.

### 3.9.1 Enter the archiving credentials in SharePoint

To automate scheduling in Capture Plus, Zetadocs needs stored credentials to archive files to your SharePoint site.

1. Create and delete a test file:

If you are using	Do this
NAV or BC Web client	<ol style="list-style-type: none"> <li>1. Create a test file.</li> <li>2. Open the Web client as the user that will run the automation.</li> <li>3. Open the Zetadocs FactBox in any page modified page, e.g. Sales Invoice page.</li> <li>4. If your credentials are not in the server, Zetadocs will display a dialog. Insert your credentials.</li> <li>5. Delete the test file from your Archiving site; to do so, delete the test file in the FactBox.</li> </ol>
Dynamics NAV Client	<ol style="list-style-type: none"> <li>1. Drop a test file to any Document Queue in the Zetadocs Server (Zetadocs Document Converter).</li> <li>2. Open the Web client as the user that will run the automation.</li> <li>3. Open the Zetadocs Document Queue page, select the Queue which contains the document and click Ok.</li> <li>4. Select the test document and click the Archive button.</li> <li>5. If your credentials are not in the server, Zetadocs will display a dialog. Insert your credentials.</li> <li>6. Close the Zetadocs Document Queue page.</li> <li>7. Delete the test file from your Archiving site and from the Document Queue if it has not already been deleted.</li> </ol>

2. Open the Development Environment and open the Object Designer.
3. Select the Codeunit tab.
4. Create a new Codeunit: File -> New.
5. Save and Compile the new Codeunit.
6. Type a name and an ID. For example:
  - Codeunit Name: Automate Document Queue,
  - ID: 51001.

### 3.9.2 Call Zetadocs Capture Interface

1. Open the Development Environment and open the Object Designer.
2. Select the Codeunit tab.
3. Select the Codeunit created in 2.5.1 and click on Design.
4. Create a new public function.
  - a. View->C/AL Globals, select the Functions tab.
  - b. Insert the name of the function, e.g. ArchiveAllLinkedRecords.
  - c. Go to the properties of the function (View->Properties) and change
    - “Local: No”.
  - d. Close the Properties dialog.
  - e. In the Functions tab of the C/AL Globals dialog, select the function you have added (e.g. ArchiveAllLinkedRecords) and click on Locals.
  - f. Add a new Parameter:
    - Name: ZetadocsDocQueueNo

- DataType: Code
- Length: 10

5. Create new Local Variables.

- Select the Variables tab.
- Add the following variables:
  - Name: ZdDocQueueItem
  - DataType: Record
  - Subtype: Zetadocs Doc. Queue Item
  
  - Name: ZdDocQueueItemTmp
  - DataType: Record
  - Subtype: Zetadocs Doc. Queue Item

Note: this is a temporary variable

- Select it and open the properties dialog (View->Properties). Change:
  - "Temporary: Yes".
  - Name: ZdCaptureInterface
  - DataType: Codeunit
  - Subtype: Zetadocs-Capture Interface

6. Close the dialogs and go back to the one that contains the code.

7. Scroll to the body of the function created previously and paste the following code:

```
//Initialize the Zetadocs Capture Interface codeunit with the Zetadocs Document Queue
(passed as a parameter)
ZdCaptureInterface.Initialize(ZetadocsDocQueueNo);

//Fetch the documents from the Document Queue on the Zetadocs Server
ZdCaptureInterface.Refresh();

//Once the documents are fetched, they exist in the Zetadocs Document Queue table.
//Get all the items related to the Zetadocs Document Queue
ZdDocQueueItem.RESET;
ZdDocQueueItem.SETRANGE("Zetadocs Doc. Queue No.", ZetadocsDocQueueNo);
IF ZdDocQueueItem.FIND('-') THEN
BEGIN
  REPEAT
    //Create a temp item for each document in the Zetadocs Document Queue
    ZdDocQueueItemTmp.DELETEALL(FALSE);
    ZdDocQueueItemTmp := ZdDocQueueItem;
    ZdDocQueueItemTmp.INSERT;
    //Archive the document if it has a link
    IF (NOT ZdCaptureInterface.ArchiveLinkedRecord(ZdDocQueueItemTmp)) THEN
      BEGIN
        //There is an error archiving the item
        //Check if the entire batch should fail
        IF (ZdCaptureInterface.GetStopBatchProcessing()) THEN
          BEGIN
            //Display corresponding error
            //Comment the following line if you do not want to stop the processing of the
batch
```

```
        ERROR('Stop batch processing is enabled. Queue: %1. Document %2 failed: %3',
ZetadocsDocQueueNo, ZdDocQueueItemTmp."File Name",
ZdCaptureInterface.GetLastErrorMessage());
        END;
    END;
    UNTIL ZdDocQueueItem.NEXT = 0;
END;

//Release the Capture Interface
ZdCaptureInterface.Clear();
```

8. Close the dialogs, save and compile the Codeunit.

### 3.9.3 Test your system

In order to test this example, you need to call the function as follows:

```
ArchiveAllLinkedRecords(<ZetadocsDocumentQueueNo>);
```

Where <ZetadocsDocumentQueueNo> is the Zetadocs Document Queue Number. For example: ZDQ0002.

Please ensure that everything works as expected before deploying it live.

Tip: An easy way to test your system is to:

1. Add the previous call to the OnRun trigger of the Codeunit that has been created in 2.5.2.
2. Save and compile the Codeunit.
3. Run it from the Development Environment.

#### Alternative ways to test

There are alternative ways to test this example depending on your needs or requirements. For example, you can call this Codeunit from the Job Queue or you may expose the Codeunit as a Web Service.

Note: if you are using Job Queues you may need to repeat the Prerequisite (at the beginning of this article) and you may need to login as the user that is running the Job Queue. Please remember that Job Queues run the OnRun trigger, so you may need to add a call to ArchiveAllLinkedRecords in this trigger.



## 5. Further Resources

To keep this guide as simple as possible we have focused on the standard installation requirements. The links below offer access to installation support materials and some more common pieces of non standard installation advice.

[Latest Advice and Resources](#) - Provides access to the latest technotes, updates and tools to support your installation, we recommend you view this page to ensure you are working with the most up to date software and advice. Also provides general advice and specific advice for special install scenarios, such as Upgrades, Group Policy and Remote Desktop Services installations.

### 4.1 How to add the Zetadocs Document FactBox to NAV or BC records outside Sales and Purchasing

This section describes how to add the Zetadocs Documents FactBox to archive and retrieve documents for non Sales or Purchase records in NAV or BC. It focuses on the following scenarios:

1. I need to add the FactBox to pages outside Sales and Purchase (e.g. Service Quotes and Service Orders)
2. I need to add the FactBox to line items (e.g. Journal entries)
3. I need to see additional related documents in the FactBox on these pages

Additional scenarios may require Equisys professional services and should be discussed separately.

#### Topics

- Introduction
- Add Zetadocs Record Mappings to Service Quotes and Service Orders
  - How to add the FactBox with Metadata to pages e. g. Service Quotes and Service Orders
  - How to add the FactBox to items which does not have a page e. g. Service Item Line
  - How to see additional documents in the FactBox:
    - Relate Service Item Lines to Service Orders
    - Relate Service Quotes to Service Orders
- How to add the FactBox to the General Ledger Entries page
- How to see additional documents in Sales and Sales Invoices

#### 4.1.1 Introduction

Zetadocs has the ability to search for documents archived against a particular NAV or BC record and display the results in the documents FactBox. By default this maps records in the Sales and Purchase areas of NAV or BC. Through configuration, it also has the ability to search for documents archived against related NAV or BC records. This is configured in the Zetadocs Record Mapping Header and Zetadocs Record Mapping Line tables. These tables

contain a collection of mappings that tell the Zetadocs search code which records are related and how. The only requirement is that there is enough data in the NAV or BC table to be able to find the related records.

When configuring the Zetadocs Mappings you will need to know the table numbers that you are referring to and the field numbers containing the data which allows you to get to the related records.

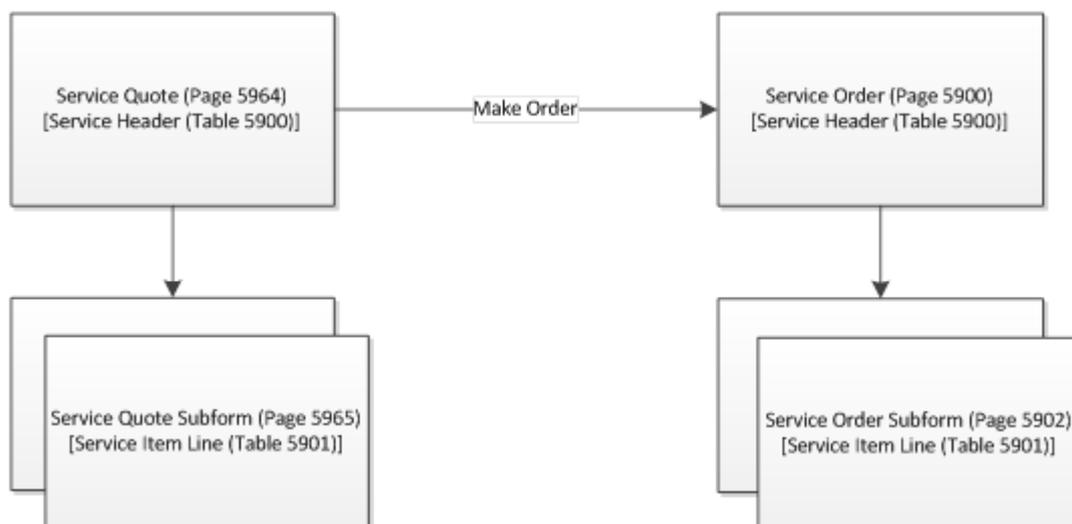
There are different mapping types:

- **Static** - This type of mapping should be used when the target record may no longer exist in the database. Zetadocs search uses the field values provided to find documents archived against the target record.
- **Live** - This type of mapping should be used for records that will always exist in the database. Zetadocs search uses the key field values to get the target record. Documents archived against this record will be found and documents archived against any records related to this record will also be found.
- **Child** - This type of mapping should be used when there will be multiple related records found in the same table. Zetadocs search will find documents archived against any of these records or against any record related to these records.

## 4.1.2 Add Zetadocs Record Mappings in Service Quotes and Service Orders

This walkthrough demonstrates how to add the Zetadocs Record Mappings into the Service Quote and Service Order pages in the NAV or BC workflow. It first illustrates how to add the Zetadocs FactBox to both the Service Quote and the Service Order pages. Secondly, it describes how to add the Zetadocs FactBox to a Service Item Line. Finally, it shows how to create the corresponding mappings so then documents attached in the Service Item Lines can be displayed in the Service Order as well as documents related to the Service Quotes.

The diagram below shows the workflow between the Service Quote and the Service Order. Please note that once the Service Order has been placed, the related Service Quote is removed from the database.



## 4.1.2.1 Add the FactBox with Metadata to pages e. g. Service Orders and Service Quotes

### How to add the FactBox with Metadata to pages e. g. Service Orders & Service Quotes

Zetadocs requires several existing NAV or BC interfaces to be modified to add Zetadocs functionality, to make this process quicker and easier we have provided the Zetadocs Interface Modification tool. This can update NAV or BC objects so that they can integrate with Zetadocs, the tool will inform you if there are any issues, for example if the objects have too extensive a set of existing customizations, in which case [manual steps are available](#). Please consult the Zetadocs Essentials Installation Guide if you require further information.

Please follow these steps:

1. Export the Service Quote and the Service Order pages in plain text from NAV or BC.
2. Make a backup of the NAV or BC Objects exported.
3. Run the Zetadocs Interface Modification Tool for the Service Quote and the Service Order, or apply the manual steps.
4. Import and compile the Service Order (modified) and Service Quotes (modified) objects in NAV or BC.

When the steps above have been completed the Zetadocs FactBox appears in the Service Quote and in the Service Order pages.

There is an optional step to add metadata information to the document archived using Zetadocs. Please visit the [ZTN4288](#) technote for further information. If you intend to include additional metadata to your Service Header Table, please run the Zetadocs Metadata Mappings table and fill it with the records displayed in the image below:

Table ID	Table Name	Mapping T...	Metadata Field Name	Metadata Fi...	Table Field ID	Fixed Value
5900	Service Header	FIXED	ZetadocsDocumentType	Text	0	Service
5900	Service Header	FIELD	ZetadocsOrganization	Text	79	
5900	Service Header	FIELD	ZetadocsRecordNo	Text	3	
5900	Service Header	CUSTOM	ZetadocsRecordType	Text	0	

Note: in the example we have included the document type, the organization, the record number and the record type; but this example could be extended to add the metadata that you may need.

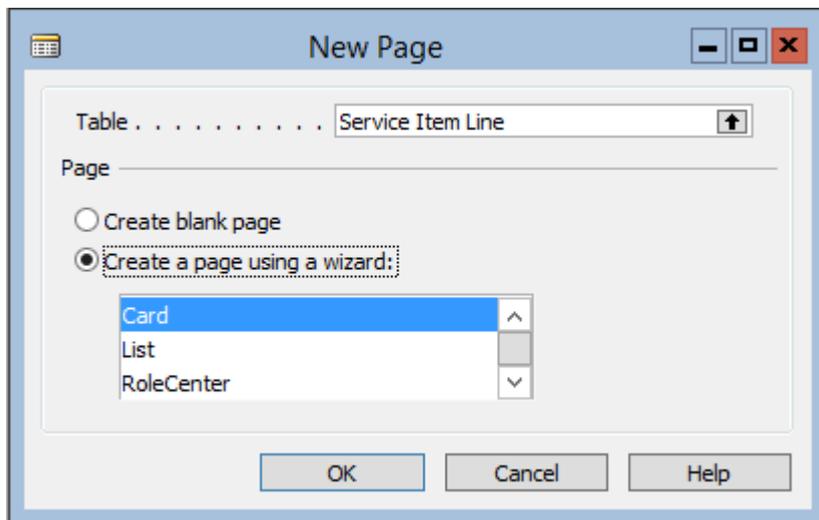
The image below displays the differences between a document archived without metadata (left) and a document archived after applying the Zetadocs Metadata Mappings in the previous step (right).

Name	Zetadocs Test document(1).txt	Name	Zetadocs Test document(2).txt
Title	Zetadocs Test document(1)	Title	Zetadocs Test document(2)
Record Type		Record Type	Zetadocs
Record No.		Record No.	SQ000002
Organization		Organization	Selangorian Ltd.
From		From	
Bcc		Bcc	
Cc		Cc	
Date Received		Date Received	
Date Sent		Date Sent	
Email		Email	
Fax		Fax	
Company No.		Company No.	
Printer		Printer	
Recipient Name		Recipient Name	
Document Type	Other	Document Type	Service
Zetadocs Archive ID		Zetadocs Archive ID	
Content Type: Zetadocs		Content Type: Zetadocs	

### 4.1.2.2 Add the FactBox to items which does not have a page e. g. Service Item Lines

This section describes how to create a new Service Item Lines page that is opened when drilling down in a specific Service Item Line and how to add the Zetadocs FactBox to it.

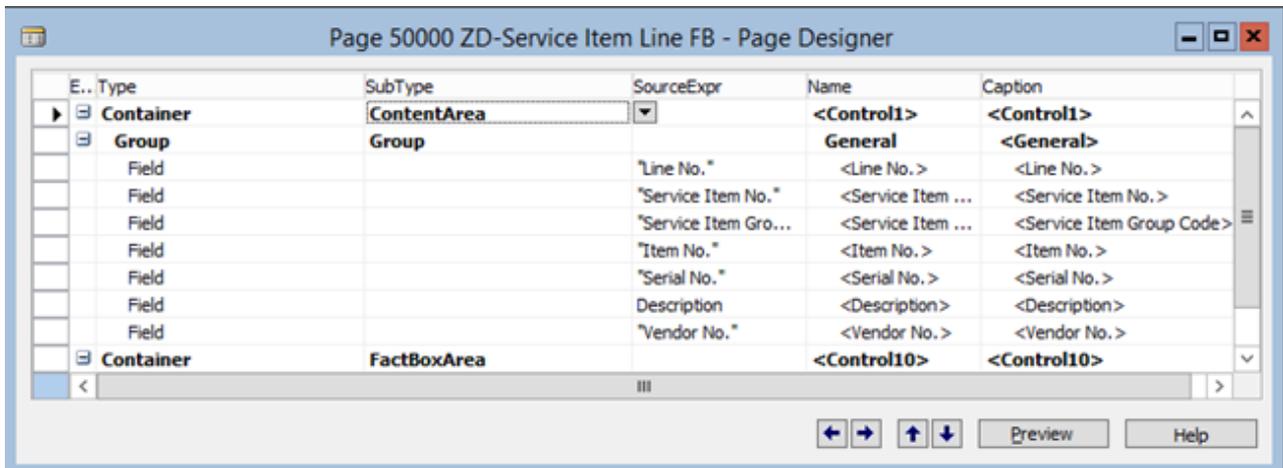
1. Create a new page in the NAV or BC Object Designer as follows:



2. Select the General FastTab and click next.
3. Select the required fields. In this example the following fields are selected: Line No. Service Item No., Service Item Group Code, Item No., Serial No., Description and Vendor No.; Click on finish.

Note: Steps 4 and 5 are only needed to add the Document FactBox with the Interface Modification Tool, otherwise please skip them.

4. Add a FactBox area to the page:



Note: the FactBoxArea does not need to be indented.

5. Add a new action container:
  - a. View>Page Actions.
  - b. Change the SubType to ActionItems, leave the defaults.
  - c. Close the Page Action Designer.
6. Save and compile your page. In this example, the ID is 50000 and the Name is ZD-Service Item Line FB.
7. Close the page.
8. Go back to the Object Designer view in NAV or BC and export the page that you have created.
9. Add the Zetadocs FactBox according to the instructions at the beginning of this section.

At this point, there should be a new page created that includes the Zetadocs FactBox. Although the Service Order has Service Item Lines, they are not directly linked to the Service Order, they use the Service Order Subform page instead. The next step is to link the ZD-Service Item Line FB page to the Service Order Subform, please follow the steps to do it:

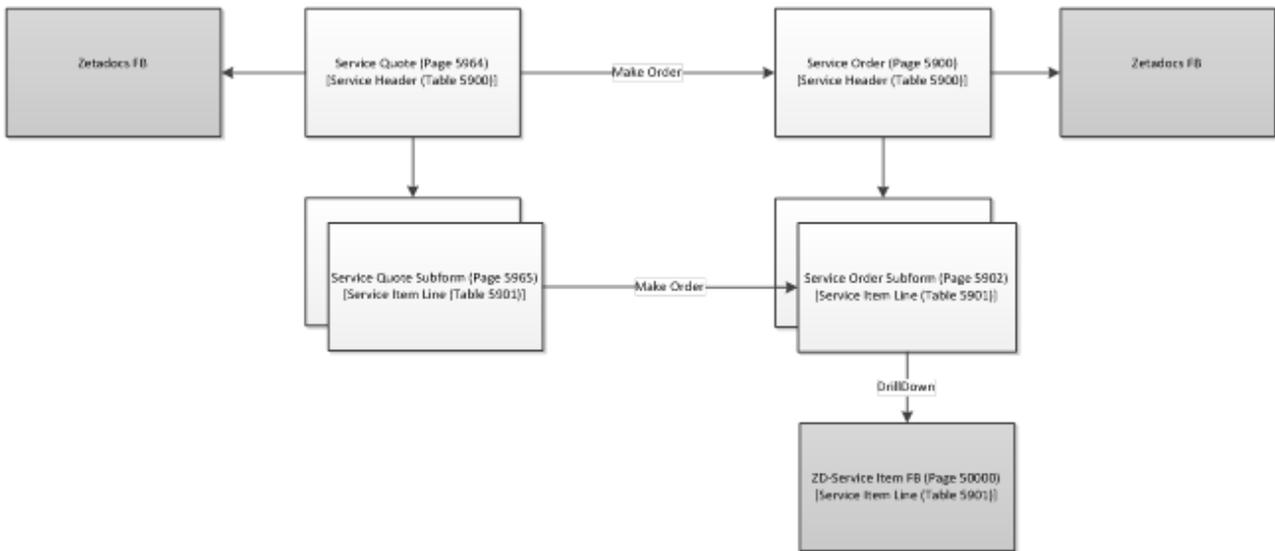
1. Open the Design view of the Service Order Subform page.
2. Open the Code Window. View>C/AL Code.
3. Scroll down to the Description – OnDrillDown() function.
4. Add the following code:  
`PAGE.RUNMODAL(50000, Rec);`  
 Where 50000 is the ID of the page created in the previous steps.
5. Close the Code Window.
6. Save and Compile the page.

The next time that the Service Order page is open, it will display the drilldown button (...) in the description field. By clicking it, it will display the ZD-Service Item Line FB page with the Zetadocs FactBox on it.

### 4.1.2.3 See additional documents in the FactBox

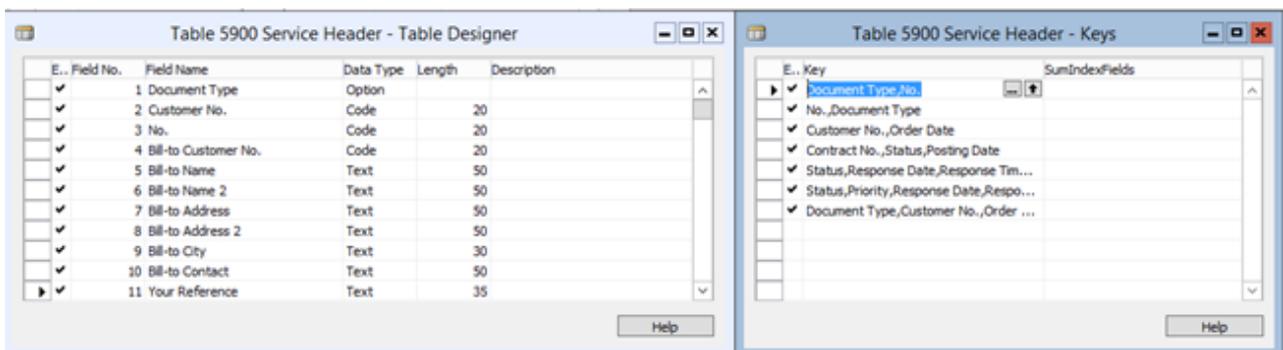
The fact that there are two FactBoxes one in the Service Order and another one in the Service Item Line does not mean that documents are automatically displayed in each other FactBoxes. The first walkthrough explains how to see the documents attached to the Service Item Line in the Service Order. This second walkthrough shows how to see related documents attached to a Service Quote in the Service Order.

The following diagram shows how Service Quotes and Service Orders are related in NAV or BC, as well as their Service Item Lines. Service Quote (Page 5964) and Service Order (Page 5900) pages are built on the top of the same Table which is the Service Header (Table 5900) whereas Service Order page uses the Service Order Subform page (Page 5902) which is built on the top of the Service Item Line Table (Table 5901).



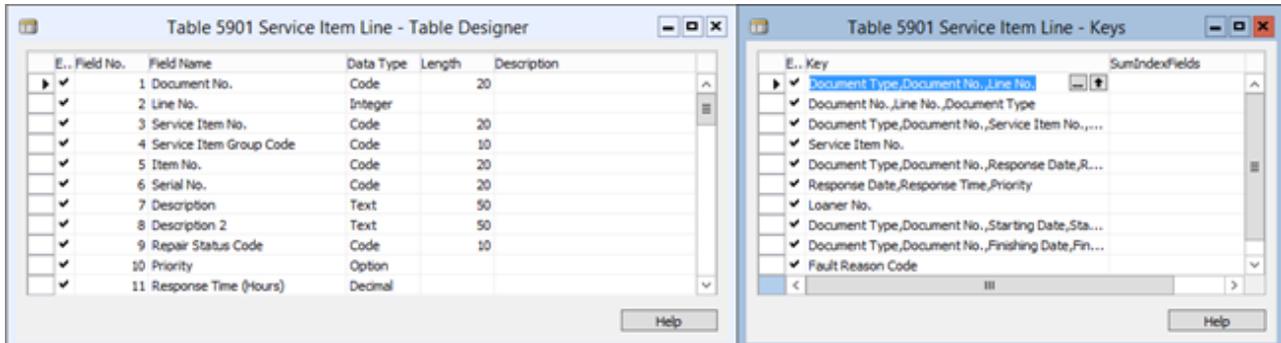
#### 4.1.2.3.1 Relate Service Item Lines to Service Orders

The reader should understand that each record in NAV or BC contains a key to uniquely identify it. It is normally composed of one or more fields. Please have a look at the keys available for the Service Header table:



The previous image displays the Design view of the Service Header table (left) with its own keys (right). It means that Zetadocs needs to know the Document Type and the No. to collect the documents archived.

In a similar way, the following image shows the keys available for the Service Item Line table:



The previous image displays the Design view of the Service Item Line table (left) with its own keys (right). Where the key is the combination of the Document Type, the Document No., and the Line No.

To create the Zetadocs Record Mapping please follow these steps:

1. Open the Object Designer in NAV or BC.
2. Run the Table 9041213 – Zetadocs Record Mappings Header and create a new rule:

Zetadocs Record Mapping Header \* Type to filter (F3) Mapping ID

No filters applied

Mappi...	Mapping T...	Source Table	Source Table Name	Target Table	Target Table Name	Mapping Description	Navi...
52	Child	114	Sales Cr.Memo Header	6660	Return Receipt Header	Posted Sales Credit Memo -> Posted R...	<input type="checkbox"/>
53	Child	5900	Service Header	...	5901 Service Item Line	Service Header -> Service Item Line	<input type="checkbox"/>

Type a new Mapping ID, the next available in this example is 53. The Mapping Type is Child in this case because it creates a Header -> Line relationship which is a one-to-many relationship. In the example the Source Table is 5900 (Service Header). The Source Table Name should be filled automatically. The Target Table corresponds with the Service Item Line table which is 5901. The Target Table Name would be completed automatically. It is highly recommended to type a description that summarizes the relationship.

3. Run the Table 9041214 – Zetadocs Record Mappings Line and relate the rule with its keys:

Zetadocs Record Mapping Line

Type to filter (F3) | Mapping ID ▼ | ➔

No filters applied

Mappi...	▲	Line ID	▲	Source Field	Target Field	Target Value
52		1		6601	6601	
53		1		1	43	
53		2		3	1	

There are two records created for the same rule (Mapping ID = 53):

- The first one (Line ID = 1) represent the first value of the Key, in our example the Source Field = 1 corresponds with the Field ID = 1 (Document Type) in the Service Header table. And the Target Field = 43 corresponds with the Field ID = 43 in the Service Item Line table (Document Type). So it means that the first key in the Service Header has a representation in the Service Item Line.
- The second one (Line ID = 2) represents the second value of the Key, in our example the Source Field = 3 corresponds with the Field ID = 3 in the Service Header table (No.). And the Target Field = 1 corresponds with the Field ID = 1 in the Service Item Line table (Document No.).

Note: This is a one-to-many relationship (Chid Mappings Type) which means that there is one missing field to retrieve an unique document, the Line No. Zetadocs will retrieve all the documents that match at least the first two keys set. As a result, it gets all the documents related to each of the lines.

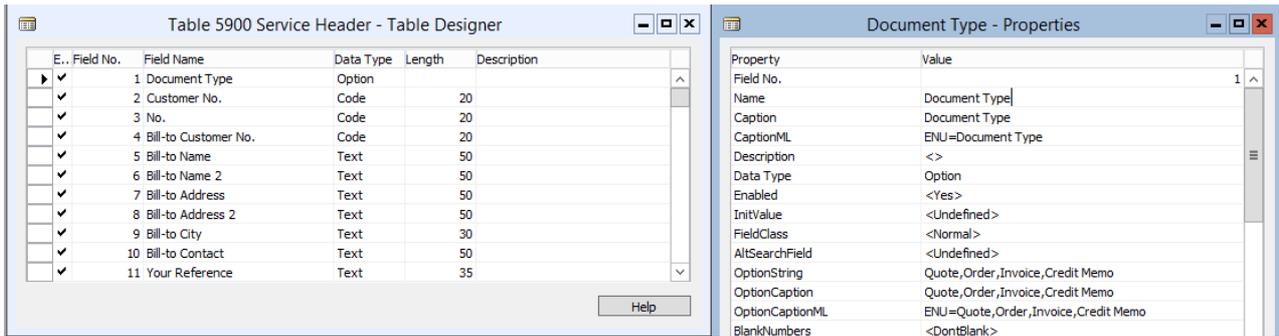
4. Close the tables and open the Service Order.

After adding the rule above with its key values, Zetadocs searches for all documents archived to the Service Item Line and displays them in the Service Order page.

### 4.1.2.3.2 Relate Service Quotes to Service Orders

This scenario explains how to add the documents attached to the Service Quote to the Service Order Zetadocs FactBox. It is worth mentioning that once a service order is created from the Service Quote page, NAV or BC deletes the quote and creates a new Service Order record. This is significant because there is some missing information that belongs to the key that we need to infer.

As you may know the option Data Type in NAV or BC could be seen as a collection, these values can be displayed in the Properties view. For example, the Document Type has four different values = {Quote, Order, Invoice, Credit Memo}. NAV or BC enumerates the values starting with zero, as a result: {Quote = 0, Order = 1, Invoice = 2, Credit Memo = 3}.



The previous image shows the Design view of the Service Header table and detailed properties of the Document Type, which contains the possible values for that type (displayed in the OptionString property).

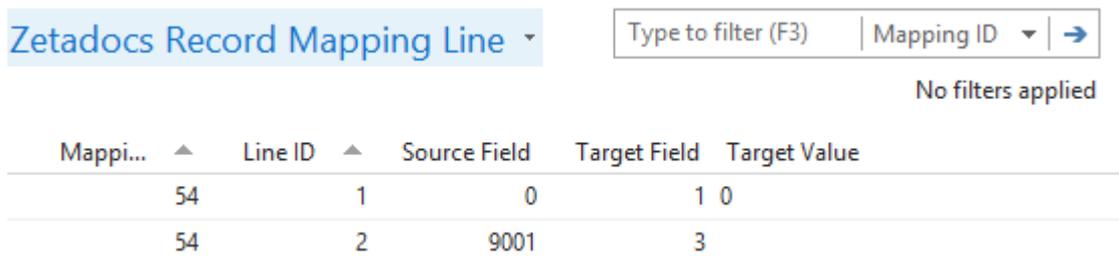
To create the Zetadocs Record Mapping please follow the steps:

1. Open the Object Designer in NAV or BC
2. Run the Table 9041213 – Zetadocs Record Mappings Header and create a new rule:



Type a new Mapping ID, the next available in this example is 54. The Mapping Type is Static in this case because the Service Quote has been deleted. In the example, the Source Table is 5900 (Service Order). The Source Table Name should be filled automatically. The Target Table corresponds with the Service Header which is 5900 (Service Quote). The Target Table Name would be completed automatically. It is highly recommended to type a description that summarizes the relationship.

3. Run the Table 9041214 – Zetadocs Record Mappings Line and relate the rule with its keys:



There are two records created for the same rule (Mapping ID = 54):

- The first one (Line ID = 1) represent the first valule of the Key, in our example the Source Field = 0 means that the mapping is not in the table, it is resolved using the Target Value. The Target Field = 1 is the Field ID = 1 (Document Type) in the Service Header table. The Document Type is an option Data Type, therefore values are in following range {Quote = 0, Order = 1, Invoice = 2, Credit Memo = 3}. So that, the Target Value = 0 means that it is a Quote.

- The second one (Line ID = 2) represents the second value of the Key, in our example the Source Field = 9001 corresponds with the Field ID = 9001 in the Service Header table (Quote No.). And the Target Field = 3 corresponds with the Field ID = 3 in the Service Header table (No.).

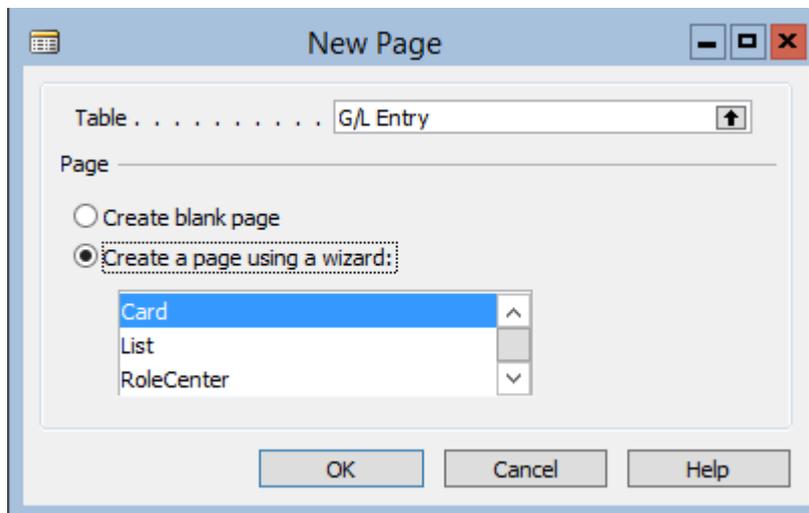
4. Close the tables and open the Service Order.

After adding the rules above Zetadocs will search for documents archived against the Service Quote and displays them in the Service Order page.

### 4.1.3 Add the Zetadocs FactBox to the General Ledger Entries page

Adding the Zetadocs FactBox to the General Ledger Entries (Page 20) is very similar to the Service Item Line example. It requires a new page to allocate the Zetadocs FactBox and link the page to the corresponding entry.

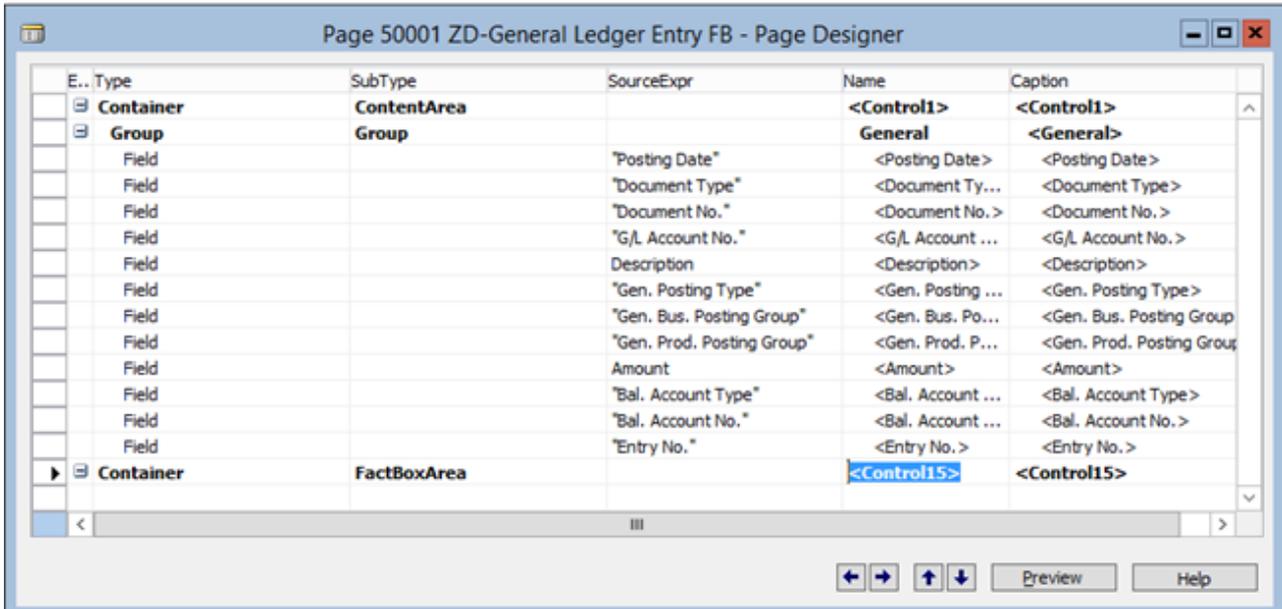
1. Create a new page in the NAV or BC Object Designer as follows:



2. Select the General FastTab and click next.
3. Select the required fields. In this example the following fields are selected: Posting Date, Document Type, Document No., G/L Account No., Description, Gen. Posting Type, Gen. Bus. Posting Group, Gen. Prod. Posting Group, Amount, Bal. Account Type, Bal. Account No. and Entry No.

Note: Steps 4 and 5 are only needed to add the Document FactBox with the Interface ModificationTool, otherwise please skip them.

4. Add a FactBox area to the page:



Note: the FactBoxArea does not need to be indented.

5. Add a new action container:
  - a. View>Page Actions.
  - b. Change the SubType to ActionItems, leave the defaults.
  - c. Close the Page Action Designer.
6. Save and compile your Page. In this example, the ID is 50001 and the Name is ZD-General Ledger Entry FB.
7. Close the page.
8. Go back to the Object Designer in NAV or BC and export the page that you have created.
9. Add the Zetadocs FactBox according to the instructions in at the beginning of this section.

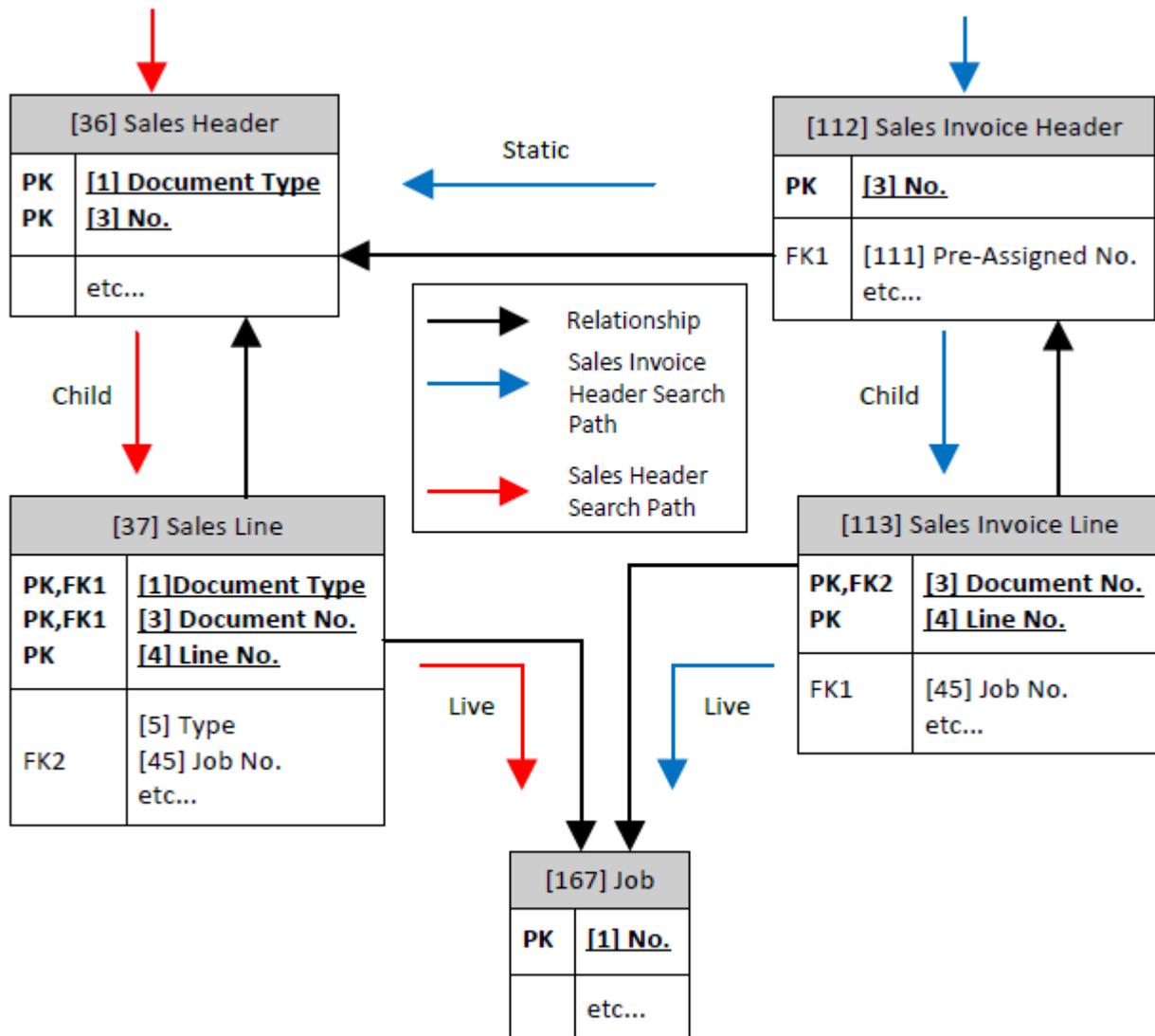
At this point, there should be a new page created that includes the Zetadocs FactBox but it is not linked to the General Ledger Entries page yet, please follow the steps to make it:

1. Open the Design view of the General Ledger Entries page.
2. Open the Code Window. View>C/AL Code.
3. Scroll down to the Description – OnDrillDown() function.
4. Add the following code:  
`PAGE.RUNMODAL(50001, Rec);`  
 Where 50001 is the ID of the page created in the previous steps.
5. Close the Code Window.
6. Save and Compile the page.

The next time that the General Ledger Entries page is open, it will display the drilldown button (...) in the description field. By clicking it, it will display the ZD-General Ledger Entry FB page with the Zetadocs FactBox on it.

### 4.1.4 See additional documents related to Jobs in Sales Invoices and Posted Sales Invoices

The diagram below shows how the search works when finding records related to a Sales Header or Sales Invoice Header record and the mapping types that are used at each stage. This example adds the required rules to get documents attached to a Job table from Sales Line or Sales Invoice Line. It also includes the rules to display documents attached to the lines in Sales Header or Sales Invoice Header records. Please note that the principle can be used to add similar functionality to areas other than sales and purchase posting routines throughout NAV or BC.



This example adds the required rules to get documents attached to a Job table from Sales Line or Sales Invoice Line. It also includes the rules to display documents attached to the lines in Sales Header or Sales Invoice Header

records. Please note that the principle can be used to add similar functionality to areas other than sales and purchase posting routines throughout NAV or BC.

Please note that you may need to add the Zetadocs Document FactBox to the Job Card, Sales Invoice and Posted Sales Invoice pages in order to test your system, if you require further information please follow the steps at the beginning of this section.

To create the Zetadocs Record Mapping please follow these steps:

1. Open the Object Designer in NAV or BC.
2. Run the Table 9041213 – Zetadocs Record Mappings Header and create a new rule:

Mapping ID	Mapping Type	Source Table	Target Table	Mapping Description
1	Static	112	36	Posted Sales Invoice -> Sales Invoice
2	Child	36	37	Sales Header -> Sales Line
3	Child	112	113	Sales Invoice Header -> Sales Invoice Line
4	Live	37	167	Sales Line -> Job
5	Live	113	167	Sales Invoice Line -> Job

In this example there are 5 new mappings, please note that you should create new mappings starting on the next free Mapping ID that you have available.

3. Run the Table 9041214 – Zetadocs Record Mappings Line and relate the rule with its keys:

Mapping ID	Line ID	Source Field	Target Field	Target Value
1	1	0	1	2
1	2	111	3	
2	1	1	1	
2	2	3	3	
3	1	3	3	
4	1	45	1	
5	1	45	1	

There are two records created for the same rules related to the Mapping ID = 1 and Mapping ID = 2, and one record for each of the other rules. Please fill the table with the values provided in the Zetadocs Mappings Line image above. Please note that Mappings IDs may be different in your system and they need to match the rules created in the previous step.

4. Close the tables.
5. Open the Posted Sales Invoice page to test your system.

After adding the rules above with its key values, the Posted Sales Invoice page searches for:

- Documents archived to the Sales Invoice and to each Sales Invoices Lines.
- Documents archived to the Posted Sales Invoices and to each Posted Sales Invoice Line.
- Documents archived to each Job in both the Sales Invoices Lines and the Posted Sales Invoice Lines.

## 4.2 Troubleshooting

Most Zetadocs components can be configured to produce their own detailed log files. These are intended to help developers working with the Zetadocs Software Development Kit or technical support staff from Equisys and other companies to resolve related problems that may occur.

You can enable the logging from each different component independently, for example you could open the Zetadocs General Settings page in your NAV or BC client and set the Log Level to Debug or you could enable the Zetadocs PDF Client logging by setting Debug in "Tools>Options...>Diagnostics".

Please note the logs are normally generated in the temporary folder of the user that is running the application or service. Zetadocs has its own log in NAV or BC, you could find it in the server where NAV or BC is installed under the service folder. For example "C:\ProgramData\Microsoft\Microsoft Dynamics NAV\80\Server\MicrosoftDynamicsNavServer\$DynamicsNAV80\users", where "MicrosoftDynamicsNavServer\$DynamicsNAV80" is the instance of NAV or BC which is running.

Contact Equisys technical support for further information.