

Answers to Developer questions

Oliver Joest

August 2016

Inhalt

- 1 Purpose
- 2 Development Environment
 - 2.1 Essentials
 - 2.2 Required Basic Knowledge
 - 2.3 Information Channels
 - 2.3.1 HG / Mercurial
 - 2.3.2 TeamCity server
 - 2.3.3 L-mobile projects
 - 2.3.4 Socialcast
 - 2.3.5 Skype
- 3 Source Control
 - 3.1 Good to Know
 - 3.1.1 General rules of thumb
 - 3.2 Concepts
 - 3.2.1 Releases
 - 3.2.2 Release Targets
 - 3.3 Branching
 - 3.4 Branches for the main development cycle
 - 3.4.1 Default
 - 3.4.2 Stable
 - 3.4.3 Feature Branches
 - 3.5 Branches for the customer development cycle
 - 3.5.1 Customer Default Branches
 - 3.5.2 Customer Stable Branches
 - 3.5.3 Customer Feature Branches
 - 3.6 Other Branches
 - 3.6.1 Experimental Branches
 - 3.6.2 Merge Branches
 - 3.6.3 Bugfix Branches
 - 3.7 Operations
 - 3.7.1 Commit
 - 3.7.2 Merge
 - 3.7.3 Graft
 - 3.8 Roles
 - 3.8.1 Maintainers
 - 3.8.2 Useful information for maintainers
 - 3.8.3 Developers
 - 3.8.4 Project Managers
 - 3.9 Common Scenarios
 - 3.9.1 Developing a feature
 - 3.9.2 FAQ
 - 3.9.3 Development for Customers
- 4 Concepts of development

- 4.1 E-Mail Dropbox
 - 4.1.1 SMTP Dropbox
 - 4.1.2 POP3 Dropbox
 - 4.1.3 Recommendation
- 4.2 Database Versioning
 - 4.2.1 Create
 - 4.2.2 Test data
 - 4.2.3 Modify aka. Migrations
 - 4.2.4 Buildserver Integration
 - 4.2.5 References
- 4.3 Logging
 - 4.3.1 log4net - SmtppAppender
- 4.4 Testing - Towards reliable state
 - 4.4.1 Selenium Tests
 - 4.4.2 Things to Know Before Writing Tests
 - 4.4.3 Unit Test Data
- 4.5 Translations
 - 4.5.1 How to Add a Translation
 - 4.5.2 How to Use a Translation
 - 4.5.3 Things Not to Forget
- 4.6 Targeting both Offline and Online modes
- 5 Integrating ERP systems (Our Legacy Integration story)
 - 5.1 Architectural considerations
 - 5.2 Integrating database systems
 - 5.2.1 MS SQL Server (MS Dynamics NAV, MS Dynamics AX)
 - 5.2.2 Oracle (Infor Erp Com, Infor Erp Blending, etc.)
 - 5.2.3 Progress (ProAlpha)
 - 5.2.4 Other database systems / CSV file integration
 - 5.2.5 Programmatic import
 - 5.3 The integration folder
 - 5.4 Normalizing legacy data to known formats - the X_External story
 - 5.5 Importing from your normalized tables - Merge
 - 5.6 The two step import
 - 5.6.1 Prepare input storage
 - 5.6.2 Create indices in temporary table -> Performance boost
 - 5.6.3 Merge operation - Step 1
 - 5.6.4 Merge operation - Step 2
 - 5.7 Importing data for distributed usage, adding offline clients to the mix
 - 5.8 How to import in a transparent fashion, logging and surveillance
 - 5.8.1 Original row count
 - 5.8.2 Intermediate records
 - 5.8.3 Try Catch
 - 5.9 Triggering import scripts
 - 5.10 Common Pitfalls
- 6 Development in the customer plugin

- 6.1 Create a new customer plugin
- 6.2 Routing
- 6.3 Menus
- 6.4 ActionFilter
- 6.5 Prevent deletion of lookups
- 6.6 Events
 - 6.6.1 NoteGenerator
- 6.7 PluginRenderActions
 - 6.7.1 List of available PluginRenderActions
- 6.8 Extension Properties
- 6.9 Validation
- 6.10 Client-side validation
- 6.11 Overriding business rules
- 6.12 Background agents
- 6.13 FAQ

7 Multitenancy

- 7.1 Database
 - 7.1.1 dbo.Domain
 - 7.1.2 dbo.DomainAuthorisedDomain
 - 7.1.3 dbo.User
 - 7.1.4 dbo.EntityType
 - 7.1.5 dbo.EntityAccess
 - 7.1.6 dbo.EntityAuthData
 - 7.1.7 dbo.GrantedEntityAccess
- 7.2 Authorization Filter
- 7.3 Services and Interfaces
 - 7.3.1 NHibernate Mappings
 - 7.3.2 SetEntityAuthDataEventHandler
- 7.4 Migrations
- 7.5 Imports
 - 7.5.1 User
 - 7.5.2 Domain / Tenants
 - 7.5.3 Entities

8 Our JS globalization story

- 8.1 Why do we need globalization?
- 8.2 Reference
- 8.3 General
- 8.4 Formatting dates and time
 - 8.4.1 Parameter
 - 8.4.2 Examples
 - 8.4.3 "01.09."
 - 8.4.4 "01.09.15"
 - 8.4.5 "01.09.2015"
 - 8.4.6 "Dienstag, 1. September 2015"
 - 8.4.7 "09:08"
 - 8.4.8 "Dienstag, 1. September 2015 09:08"
- 8.5 Formatting numbers
 - 8.5.1 Parameter
 - 8.5.2 Plain Numbers
 - 8.5.3 Percentage
- 8.6 Parsing dates
 - 8.6.1 Parameters
 - 8.6.2 Examples
- 8.7 Parsing numbers
 - 8.7.1 Parameters
 - 8.7.2 Examples
- 8.8 Calendar constants

9 Moment.js for date/time operations

9.1 Preamble

9.2 Reference

9.3 Examples

9.3.1 Now

9.3.2 Add

9.3.3 Start of Time

9.3.4 Format

10 Custom knockout binding handlers

- 10.1 autosize
 - 10.1.1 Example
 - 10.1.2 Parameters
- 10.2 colorPicker
 - 10.2.1 Example
 - 10.2.2 Example with additional colorpicker options
 - 10.2.3 Parameters
- 10.3 datePicker
 - 10.3.1 Example
 - 10.3.2 Parameters
- 10.4 dateRange
 - 10.4.1 Example
 - 10.4.2 Parameters
- 10.5 dateText
 - 10.5.1 Example
 - 10.5.2 Example 2: Specifying a custom pattern
 - 10.5.3 Example 3: Use time ago format
 - 10.5.4 Parameters
- 10.6 dropzone
 - 10.6.1 Example 1: Use the dropzone to upload one file
 - 10.6.2 Example 2: Use the dropzone to upload multiple files
 - 10.6.3 Parameters
- 10.7 durationText
 - 10.7.1 Example
 - 10.7.2 Example 2: Specifying a custom template
 - 10.7.3 Parameters
- 10.8 fileInput
 - 10.8.1 Example
 - 10.8.2 Parameters
- 10.9 fileResource
 - 10.9.1 Example 1: Opening a file resource in a new tab
 - 10.9.2 Example 2: Downloading a file resource
 - 10.9.3 Parameters
- 10.10 fileSize
 - 10.10.1 Example
 - 10.10.2 Parameters
 - 10.10.3 Note
- 10.11 flotChart
 - 10.11.1 Example
 - 10.11.2 Parameters
- 10.12 fullCalendar
 - 10.12.1 Example
 - 10.12.2 Parameters
- 10.13 lookupValue
 - 10.13.1 Example
 - 10.13.2 Parameters
- 10.14 map
 - 10.14.1 Example
 - 10.14.2 Parameters
- 10.15 mapLink
 - 10.15.1 Example
 - 10.15.2 Parameters
- 10.16 money

- 10.16.1 Example
 - 10.16.2 Parameters
 - 10.16.3 Note
- 10.17 mProgress
 - 10.17.1 Example
 - 10.17.2 Parameters
- 10.18 noRequiredPermission
 - 10.18.1 Example 1
 - 10.18.2 Parameters
- 10.19 popover
 - 10.19.1 Example
 - 10.19.2 Parameters
- 10.20 requiredPermission
 - 10.20.1 Example 1
 - 10.20.2 Example 2
 - 10.20.3 Parameters
- 10.21 signaturePad
 - 10.21.1 Example
 - 10.21.2 Parameters
 - 10.21.3 Note
- 10.22 tooltip
 - 10.22.1 Example
 - 10.22.2 Parameters
- 10.23 translatedText
 - 10.23.1 Example 1
 - 10.23.2 Example 2: Translating to a specific language
 - 10.23.3 Parameters
- 10.24 translatedTitle
- 10.25 translatedAriaLabel
- 10.26 translatedDataWmlSpeechCommand
- 10.27 userAvatar
 - 10.27.1 Example
 - 10.27.2 Parameters
 - 10.27.3 Note
- 10.28 userDisplayName
 - 10.28.1 Example 1
 - 10.28.2 Example 2
 - 10.28.3 Parameters
- 10.29 select2autocomplete
 - 10.29.1 Example
 - 10.29.2 Parameters
- 10.30 text
 - 10.30.1 Example 1
 - 10.30.2 Example 2
 - 10.30.3 Example 3
 - 10.30.4 Example 4
 - 10.30.5 Parameters
- 10.31 durationInput
 - 10.31.1 Example
 - 10.31.2 Parameters
- 11 Custom knockout components

- 11.1 barcode-format
 - 11.1.1 Example
 - 11.1.2 Parameters
- 11.2 barcode
 - 11.2.1 Example
 - 11.2.2 Parameters
- 11.3 addressBlock
 - 11.3.1 Example
 - 11.3.2 Parameters
- 11.4 addressEditor
 - 11.4.1 Example
 - 11.4.2 Parameters
- 11.5 addressSelector
 - 11.5.1 Example
 - 11.5.2 Parameters
- 11.6 barcode scanner
 - 11.6.1 Example
 - 11.6.2 Parameters
- 11.7 contactData
 - 11.7.1 Example
 - 11.7.2 Parameters
- 11.8 countWidget
 - 11.8.1 Example
 - 11.8.2 Parameters
- 11.9 dateFilter
 - 11.9.1 Example
 - 11.9.2 Parameters
- 11.10 emptyStateBox
 - 11.10.1 Example
 - 11.10.2 Parameters
- 11.11 floatingActionButton
 - 11.11.1 Example
 - 11.11.2 Parameters
 - 11.11.3 Support for multiple actions
- 11.12 flotChart
 - 11.12.1 Example
 - 11.12.2 Parameters
- 11.13 formElement
 - 11.13.1 Example
 - 11.13.2 Parameters
- 11.14 fullCalendar
 - 11.14.1 Example
 - 11.14.2 Parameters
- 11.15 generic-list-selection
 - 11.15.1 Example
 - 11.15.2 Parameters
- 11.16 inline editor
 - 11.16.1 Example
 - 11.16.2 Parameters
- 11.17 miniChart
 - 11.17.1 Example
- 11.18 pmbb

- 11.18.1 Example
 - 11.18.2 Parameters
- 11.19 scaleFilter
 - 11.19.1 Example
 - 11.19.2 Parameters
- 11.20 signaturePad
 - 11.20.1 Example
 - 11.20.2 Parameters
- 11.21 timeRangeFilter
 - 11.21.1 Example
 - 11.21.2 Parameters
- 11.22 taskListBlock
 - 11.22.1 Example
 - 11.22.2 Parameters
- 11.23 statusChooser
 - 11.23.1 Example
 - 11.23.2 Parameters
- 11.24 collapsibleBlock
 - 11.24.1 Example
 - 11.24.2 Parameters
- 11.25 block
 - 11.25.1 Example
 - 11.25.2 Parameters
- 12 Custom knockout extenders
 - 12.1 filterOperator
 - 12.1.1 Example
 - 12.1.2 Parameters
- 13 Recipes for developing in Customer Plugins

13.1 Working with Customer Plugins

13.1.1 The CustomerNamePlugin.cs file

13.1.2 PluginAttribute

13.1.3 CustomerPlugin inheritance

13.2 Background services

13.2.1 Background Services - Automatic Session Handling

13.2.2 Background Services - Manual Session Handling

13.2.3 Background Services - jobs.xml

13.3 Business Rules

13.3.1 CrmModelBinder for new created classes

13.3.2 Overriding rules from Customer plugin

13.3.3 Ignoring rules from Customer plugin

13.3.4 Creating new rules from Customer plugin

13.3.5 HtmlHelperExtensions for Business rules

13.4 Routing

13.4.1 Create new Routes for your customer plugin

13.4.2 Redirecting existing routes to customer plugin

13.5 Menu extensions~~~~

13.5.1 Unregistering existing menu entries

13.5.2 Registering new menu entries

13.5.3 Menu entries, adding permissions

13.5.4 Menu entries, adding icons

13.6 Generating IDs on the client-side

13.6.1 Client Side Additions

13.6.2 Server Side Changes

13.7 Creating and querying a hybrid model

13.7.1 Step 1 - Tell the application to generate a client side model for your entity

13.7.2 Step 2 - Create a sync service

13.7.3 Step 3 - Altering your entity to support client-side ID generation

13.7.4 Step 4 - Using the generated model to query offline and online sources

14 Customizing the ServiceOrderDispatch report

14.1 Overriding & extending the ViewModel

14.1.1 JavaScript ViewModel

14.2 How to define the margins and header / footer sizes

14.3 Add & format paging indicators

14.4 Include additional CSS & JS assets

14.5 Stamping and appendices

14.6 Changing the view

15 Optimize performance for web applications

15.1 Database

15.2 Database Maintenance

15.3 IIS

15.4 Application

15.5 programming

16 Documentation with the Nudoc tool

- 16.1 Directory Layout
- 16.2 Usage
- 16.3 Flowcharts
- 16.4 Tips
- 16.5 Requirements
- 16.6 Where to find it
- 16.7 Stylesheet
 - 16.7.1 Structure
 - 16.7.2 Language and tone
 - 16.7.3 Specification
 - 16.7.4 Structure and Flow
 - 16.7.5 Screenshots
 - 16.7.6 Table of content
 - 16.7.7 Links
 - 16.7.8 German - Wiederkehrende Begriffe für einfachere Wiedererkennung
 - 16.7.9 German - Anti-Patterns

17 Deployment

- 17.1 Initial deployment
- 17.2 Moving Test to Production
- 17.3 Hotfixes in Production
- 17.4 Upgrading from release x.x to x.y

18 Documentation of application settings

- 18.1 Something you need to know for this document
 - 18.1.1 How do I identify a configuration switch as price list position?

19 Documentation of plugin settings

- 19.1 Crm.Article
- 19.2 Crm.Campaigns (CRM/sales: SW000344)
 - 19.2.1 FinishedCampaignsSelectableForDays (int)
 - 19.2.2 CampaignPriceStep (int)
- 19.3 Crm.Configurator (CRM/sales: SW000347)
- 19.4 Crm.Documentation
- 19.5 Crm.Documentation.CommandLine
- 19.6 Crm.DynamicForms
- 19.7 Crm.ErpExtension (CRM/sales: SW000531 + SW000532)
- 19.8 Crm.InforExtension (CRM/sales: SW000267; service: SW000270)
- 19.9 Crm.Offline (CRM/sales: SW000327)
- 19.10 Crm.Order (CRM/sales: SW000345, SW000346)
- 19.11 Crm.Project (CRM/sales: SW000342)
- 19.12 Crm.ProjectOrders (CRM/sales: SW000342)
- 19.13 Crm.Service (Service: SW000351)
- 19.14 Crm.VisitReport (CRM/sales: SW000340; SW000341)
- 19.15 Main (CRM/sales: SW000325; Service: SW000351)
- 19.16 Main.SmtpDropbox (CRM/sales: SW000325; Service: SW000351)
- 19.17 Sms.Checklists (service: SW000373)
- 19.18 Sms.TimeManagement (service: SW000367)

20 Documentation of configuration settings

20.1 Something you need to know for this chapter

20.1.1 How to create "string arrays"

20.2 Main

- 20.2.1 Address/DisplayOnlyRegionKey (bool)
- 20.2.2 AllowCompanyTypeSelection (bool)
- 20.2.3 ApplePushNotification/CertificateFileName (string)
- 20.2.4 ApplePushNotification/CertificatePassword (string)
- 20.2.5 ApplePushNotification/ProductionEnvironment (bool)
- 20.2.6 CefToPdfPath (string)
- 20.2.7 Configuration/BravoActiveForCompanies (bool)
- 20.2.8 Configuration/BravoActiveForPersons (bool)
- 20.2.9 CompanyGroupFlags/AreSearchable (bool)
- 20.2.10 CompanyNolsGenerated(bool)
- 20.2.11 CompanyNolsCreateable(bool)
- 20.2.12 CompanyNolsEditable(bool)
- 20.2.13 Cordova/AndroidAppLink (string)
- 20.2.14 Cordova/AppeliosAppLink (string)
- 20.2.15 Cordova/Windows10AppLink (string)
- 20.2.16 DropboxForwardPrefixes (string)
- 20.2.17 DropboxDomain (string)
- 20.2.18 MinFileSizeInBytes (int)
- 20.2.19 MinPasswordStrength (int)
- 20.2.20 DropboxLogMessages (bool)
- 20.2.21 FileResource/AllowedContentTypes (string[])
- 20.2.22 FileResource/ContentTypesOpenedWithoutSandbox (string[])
- 20.2.23 Geocoder/GeocoderService (string)
- 20.2.24 Geocoder/BingMapsApiKey (string)
- 20.2.25 Geocoder/GoogleMapsApiKey (string)
- 20.2.26 Geocoder/MapQuestApiKey (string)
- 20.2.27 Geocoder/YahooMapsApiKey (string)
- 20.2.28 Geocoder/YahooMapsApiSecret (string)
- 20.2.29 Lucene/LegacyNamelsDefault (bool)
- 20.2.30 MapTileLayerUrl (string)
- 20.2.31 PasswordReset/ExpirationInMinutes (int)
- 20.2.32 PasswordReset/MaxEmailsPerHour (int)
- 20.2.33 Person/BusinessTitlesLookup (bool)
- 20.2.34 Person/DepartmentIsLookup (bool)
- 20.2.35 PersonNolsGenerated(bool)
- 20.2.36 PersonNolsCreateable(bool)
- 20.2.37 PersonNolsEditable(bool)
- 20.2.38 RedisConfiguration (string)
- 20.2.39 Report/HeaderHeight (double)
- 20.2.40 Report/HeaderSpacing (double)
- 20.2.41 Report/FooterHeight (double)
- 20.2.42 Report/FooterSpacing (double)
- 20.2.43 Site/HostEditable (bool)
- 20.2.44 Site/PluginsEditable (bool)
- 20.2.45 StripLeadingZerosFromLegacyId (bool)
- 20.2.46 UseActiveDirectoryAuthenticationService (bool; CRM/sales: SW000336; Service: SW000366)
- 20.2.47 ActiveDirectoryEndpoint
- 20.2.48 Maintenance/PasswordResetTokenDeprecationDays (int)
- 20.2.49 Maintenance/PostingDeprecationDays (int)
- 20.2.50 Maintenance/MessageDeprecationDays (int)
- 20.2.51 Maintenance/LogDeprecationDays (int)
- 20.2.52 Maintenance/ErrorLogDeprecationDays (int)
- 20.2.53 Maintenance/ReplicatedClientDeprecationDays (int)
- 20.2.54 Maintenance/FragmentationLevel1 (int 0-100)
- 20.2.55 Maintenance/FragmentationLevel2 (int 0-100)
- 20.2.56 Maintenance/CommandTimeout (int)

20.3 Main

- 20.3.1 Posting/MaxRetries (int)
- 20.3.2 Posting/RetryAfter (int)

20.4 Main.SmtpDropbox

- 20.4.1 SmtpDropboxAgent/Port (int)

20.5 Crm.Campaigns

- 20.5.1 CampaignNolsGenerated(bool)
- 20.5.2 CampaignNolsCreateable(bool)
- 20.5.3 CampaignNolsEditable(bool)

20.6 Crm.ErpExtension

- 20.6.1 EnableAddressExport (bool)
- 20.6.2 EnableCompanyExport (bool)
- 20.6.3 EnableCommunicationExport (bool)
- 20.6.4 EnablePersonExport (bool)
- 20.6.5 ErpSystemID (string)
- 20.6.6 ErpSystemName (string)
- 20.6.7 ObjectLinkIntegration (string)
- 20.6.8 TurnoverCurrencyKey (string)

20.7 Crm.InforExtension

- 20.7.1 InforExport/InforErpComVersion (string)
- 20.7.2 InforExport/ShortFieldNames (bool)

20.8 Crm.Order

- 20.8.1 OffersEnabled (bool) (CRM/sales: SW000345, SW000346)
- 20.8.2 OrderBarcodeEnabled (bool)
- 20.8.3 OrderBillingAddressEnabled (bool)
- 20.8.4 OrderComissionEnabled (bool)
- 20.8.5 OrderDeliveryAddressEnabled (bool)
- 20.8.6 OrderItemDiscountEnabled (bool)
- 20.8.7 OrderPrivateDescriptionEnabled (bool)
- 20.8.8 OrderSignatureEnabled (bool)
- 20.8.9 Order/OrderNolsGenerated(bool)
- 20.8.10 Order/OrderNolsCreateable(bool)
- 20.8.11 Order/OrderNolsEditable(bool)
- 20.8.12 Offer/OfferNolsGenerated(bool)
- 20.8.13 Offer/OfferNolsCreateable(bool)
- 20.8.14 Offer/OfferNolsEditable(bool)
- 20.8.15 PDFHeaderMargin (double)
- 20.8.16 PDFFooterMargin (double)
- 20.8.17 PDFFooterTextPush (double)

20.9 Crm.Project

- 20.9.1 Configuration/BravoActiveForProjects (bool)
- 20.9.2 ProjectsHaveAddresses (bool)

20.10 Crm.PerDiem

- 20.10.1 Email/PerDiemReportApprovers (string)
- 20.10.2 Email/PerDiemReportRecipients (string)
- 20.10.3 Email/SendPerDiemReportToResponsibleUser (bool)
- 20.10.4 Expense/ClosedExpensesHistorySyncPeriod (int)
- 20.10.5 Expense/MaxDaysAgo (int)
- 20.10.6 PerDiemReport/ShowClosedReportsSince (int)
- 20.10.7 TimeEntry/MinutesInterval (int)
- 20.10.8 TimeEntry/AllowOverlap (bool)
- 20.10.9 TimeEntry/ClosedTimeEntriesHistorySyncPeriod (int)
- 20.10.10 TimeEntry/DefaultStart (string)
- 20.10.11 TimeEntry/MaxDaysAgo (int)
- 20.10.12 TimeEntry/DefaultWorkingHoursPerDay (int)
- 20.10.13 TimeEntry/ShowTimeZone (bool)

20.11 Crm.Project

- 20.11.1 Potential/PotentialNolsGenerated(bool)
- 20.11.2 Potential/PotentialNolsCreateable(bool)
- 20.11.3 Potential/PotentialNolsEditable(bool)
- 20.11.4 ProjectNolsGenerated(bool)
- 20.11.5 ProjectNolsCreateable(bool)
- 20.11.6 ProjectNolsEditable(bool)

20.12 Crm.Service

20.12.1 AdHoc/AdHocNumberingSequenceName (string)
20.12.2 Dispatch/DispatchNolsGenerated(bool)
20.12.3 Dispatch/DispatchNolsCreateable(bool)
20.12.4 Dispatch/DispatchNolsEditable(bool)
20.12.5 Dispatch/SuppressEmptyMaterialsInReport(bool)
20.12.6 Dispatch/SuppressEmptyExpensePostingsInReport(bool)
20.12.7 Dispatch/SuppressEmptyErrorTypesInReport(bool)
20.12.8 Dispatch/SuppressEmptyTimePostingsInReport(bool)
20.12.9 Dispatch/SuppressExpensePostingsInReport(bool)
20.12.10 Dispatch/SuppressErrorTypesInReport(bool)
20.12.11 Dispatch/SuppressEmptyJobsInReport(bool)
20.12.12 Email/ClosedByRecipientForReplenishmentReport (bool)
20.12.13 Email/DispatchReportRecipients (string array)
20.12.14 Email/ReplenishmentOrderRecipients (string array)
20.12.15 Email/SendDispatchNotificationEmails (bool)
20.12.16 Email/SendDispatchRejectNotificationEmails (bool)
20.12.17 Email/SendDispatchFollowUpOrderNotificationEmails (bool)
20.12.18 Email/SendDispatchReportsOnCompletion (bool)
20.12.19 Email/SendDispatchReportToDispatcher (bool)
20.12.20 Email/SendDispatchReportToTechnician (bool)
20.12.21 Email/SendServiceOrderReportsOnCompletion (bool)
20.12.22 Email/SendServiceOrderReportToDispatchers (bool)
20.12.23 Export/ExportDispatchReportsControlFileExtension (string)
20.12.24 Export/ExportDispatchReportsControlFileContent (string)
20.12.25 Export/ExportDispatchReportsControlFilePattern (string)
20.12.26 Export/ExportDispatchReportsFilePattern (string)
20.12.27 Export/ExportDispatchReportsOnCompletion (bool)
20.12.28 Export/ExportDispatchReportsPath (string)
20.12.29 Export/ExportServiceOrderReportsOnCompletion (bool)
20.12.30 Export/ExportServiceOrderReportsPath (string)
20.12.31 Export/ExportServiceOrderReportsUncDomain (string)
20.12.32 Export/ExportServiceOrderReportsUncPassword (string)
20.12.33 ExportServiceOrderReportsUncUser (string)
20.12.34 ReplenishmentOrder/ClosedReplenishmentOrderHistorySyncPeriod (int)
20.12.35 InstallationNolsGenerated(bool)
20.12.36 InstallationNolsCreateable(bool)
20.12.37 InstallationNolsEditable(bool)
20.12.38 Service/Dispatch/Requires/CustomerSignature (bool)
20.12.39 Service/Dispatch/Show/EmptyTimesOrMaterialsWarning (string)
20.12.40 ServiceCase/OnlyInstallationsOfReferencedCustomer (bool)
20.12.41 ServiceCase/Signature/Enable/Originator (bool)
20.12.42 ServiceCase/Signature/Enable/Technician (bool)
20.12.43 ServiceCase/ServiceCaseNolsGenerated(bool)
20.12.44 ServiceCase/ServiceCaseNolsCreateable(bool)
20.12.45 ServiceCase/ServiceCaseNolsEditable(bool)
20.12.46 ServiceContract/OnlyInstallationsOfReferencedCustomer (bool)
20.12.47 ServiceContract/MaintenanceOrderGenerationMode (string)
20.12.48 ServiceContract/CreateMaintenanceOrderTimeSpanDays (int)
20.12.49 ServiceContract/MaintenancePlan/AvailableTimeUnits (string array)
20.12.50 ServiceContract/ReactionTime/AvailableTimeUnits (string array)
20.12.51 ServiceContract/ServiceContractNolsGenerated(bool)
20.12.52 ServiceContract/ServiceContractNolsCreateable(bool)
20.12.53 ServiceContract/ServiceContractNolsEditable(bool)
20.12.54 ServiceObject/ObjectNolsGenerated(bool)
20.12.55 ServiceObject/ObjectNolsCreateable(bool)
20.12.56 ServiceObject/ObjectNolsEditable(bool)
20.12.57 ServiceOrderMaterial/CreateReplenishmentOrderItemsFromServiceOrderMaterial (bool)
20.12.58 ServiceOrderMaterial/ShowPricesInMobileClient (bool)
20.12.59 ServiceOrder/DefaultDuration (string)
20.12.60 ServiceOrder/OnlyInstallationsOfReferencedCustomer (bool)
20.12.61 ServiceOrder/GenerateAndAttachJobsToUnattachedTimePostings (bool)
20.12.62 ServiceOrder/OrderNolsGenerated(bool)
20.12.63 ServiceOrder/OrderNolsCreateable(bool)
20.12.64 ServiceOrder/OrderNolsEditable(bool)
20.12.65 ServiceOrderDispatch/ReadGeolocationOnDispatchStart (bool)
20.12.66 ServiceOrderExpensePosting/UseArticleAsExpenseType
20.12.67 ServiceOrderTimePosting/ClosedTimePostingsHistorySyncPeriod (int)

- 20.12.68 ServiceOrderTimePosting/MaxDaysAgo (int)
- 20.12.69 ServiceOrderTimePosting/ShowTechnicianSelection (bool)
- 20.12.70 ServiceOrderTimePosting/MinutesInterval (int)
- 20.12.71 ServiceOrderTimePosting/AllowOverlap (bool)
- 20.12.72 UserExtension/OnlyUnusedLocationNosSelectable (bool)
- 20.12.73 AttentionTasks/CheckAllInstallationsWarranties (bool)
- 20.13 Crm.VisitReport
 - 20.13.1 DefaultVisitTimeSpanHours (double)
 - 20.13.2 Visit/AvailableTimeUnits (string array)
- 20.14 Sms.Checklists
 - 20.14.1 Dispatch/CustomerSignatureValidateChecklists (bool)
- 20.15 Sms.Scheduler
 - 20.15.1 WorkingTime/FromDay (int)
 - 20.15.2 WorkingTime/ToDay (int)
 - 20.15.3 WorkingTime/FromHour (int)
 - 20.15.4 WorkingTime/ToHour (int)
 - 20.15.5 WorkingTime/MinutesInterval (int)
 - 20.15.6 WorkingTime/IgnoreWorkingTimesInEndDateCalculation (bool)
 - 20.15.7 ServiceOrderZipCodeAreaLength (int)
 - 20.15.8 DispatchesAfterReleaseAreEditable (bool)
 - 20.15.9 DashboardCalendar/ShowAbsencesInCalendar (bool)
- 20.16 Sms.TimeManagement
 - 20.16.1 GeolocationGetCurrentPositionTimeout (int)
 - 20.16.2 Task/AttentionTaskTypeKey (string)
- 21 API
 - 21.1 Introduction
 - 21.2 Supported Formats
 - 21.3 HTTP Request Methods
 - 21.4 Tools
 - 21.5 Authentication
 - 21.5.1 Basic Authentication
 - 21.5.2 Token Authentication
 - 21.6 User Agent
 - 21.7 Metadata
 - 21.7.1 Metadata Annotations
 - 21.7.2 Extension Values
 - 21.8 API Explorer
 - 21.9 Reading
 - 21.9.1 Single Entity
 - 21.9.2 Collection of Entities
 - 21.9.3 Filtering
 - 21.9.4 Counting
 - 21.10 Writing
 - 21.10.1 Direct Save
 - 21.10.2 Special Entities
 - 21.10.3 Creating / Updating
 - 21.10.4 Deleting
 - 21.11 Batch
 - 21.11.1 Continue on Error
 - 21.12 Return Codes
 - 21.13 Client Development
 - 21.14 Further Reading
 - 21.15 Known Limitations
 - 21.16 Appendix A
 - 21.16.1 Request
 - 21.16.2 Response
- 22 API

22.1 Attributes

22.1.1 ExplicitEntitySet

22.1.2 ExplicitExpand

22.1.3 RestrictedField

22.1.4 NotMapped

22.2 Configuration

23 API Development

23.1 Client

23.2 Attributes

23.2.1 ExplicitEntitySet

23.2.2 ExplicitExpand

23.3 RestrictedField

23.3.1 NotMapped

23.3.2 NotReceived

23.3.3 RestrictedType

23.4 Extending the API

23.4.1 Configuring the model

23.4.2 Manipulating queries

23.4.3 Manipulating results

23.4.4 Manipulating data manipulating requests

23.4.5 Manipulating keys

24 OpenId configuration

24.1 Preparation

24.2 Configuration

24.3 FAQ

24.3.1 How does the application know which user has logged in?

24.3.2 Is it necessary to create a corresponding user in the CRM?

24.3.3 What happens if a user logs in with the identity provider, but it's username is not assigned to any user in the CRM?

25 Unit Testing with Jest

25.1 Environment

25.2 Matchers overview

25.3 How to create a unit test

25.4 Tests with imports

25.5 Test with Promises

1 Purpose

This page makes an overall introduction to the CRM / service documentation. Please use the menu to navigate through the documentation. L-mobile CRM / service project is expanding with its growing codebase. However, without strict conventions on the project structure, the growth of the project also increases the chances that some code will eventually be overridden by some other while making it harder to debug the problems and understand how the program flow works.

2 Development Environment

For the development of CRM and service components, there exists a typical environment that would speed up the developers significantly.

2.1 Essentials

The typical L-mobile development environment is as follows.

These components are required to build and run the L-mobile CRM / service locally

- Microsoft Windows 7 or later (x64 is preferred and supported) (Should be installed by IT)
- Microsoft Visual Studio 2015 (Should be installed by IT)
- [Microsoft SQL Server 2014 R2 Express with Management Studio](#)
- [TortoiseHG](#)

Important These components are very important for a compatible and productive environment

- [Resharper](#) (L-mobile has a license)
 - a good editor like [Notepad++](#)
-

2.2 Required Basic Knowledge

The essential knowledge needed to be able to extend and build upon the L-mobile CRM / service project.

In the CRM project, you will typically encounter C# and Javascript code. However, familiarity with these two languages by itself would not be enough, as there are frameworks and conventions in place.

The most important frameworks that the developers need to be familiar with are [jQuery](#) and [ASP.NET MVC](#).

Meanwhile, being able to write proper [Selenium test](#) enable application developers to make sure the user interface keeps working while the application gets extended.

2.3 Information Channels

Communication Channels for Data Exchange. The channels can be used to communicate with other developers or get the latest information about the project when you need help, detailed information or feedback on development, documentation or decision making.

2.3.1 HG / Mercurial

Mercurial is used to manage versions of the source code, ease collaboration and enable reverting to older versions when necessary. TortoiseHG is used as the preferred client for connecting to the L-mobile HG repositories.

To access the L-mobile HG repositories, please enter the following url to your HG client:

<https://svn.l-mobile.com/hg/Crm>

For further information about the usage of the TortoiseHG client, please refer to its [manual](#).

You'll need an account for accessing the repositories. If you still don't have an account, please contact your lead developer.

2.3.2 TeamCity server

This software runs on the build server which re-builds the project on every commit by any developer. TeamCity is used to track the status of these builds. When you go to [Teamcity](#), you will see the list of the projects that are being tracked by TeamCity. By clicking on the name of any of those projects, you can see detailed information about the recent builds like the list of the modified files from the last build, test pass/fail status, any warnings/errors encountered during the build and the build time.

2.3.3 L-mobile projects

This software helps you keep track of the bugs assigned to you while allowing you to add some meta data, see the list of modified files and comment on them.

Important It's never a "bug", it's always an "issue"

You can directly go to [the Bug Tracker](#) and use the filtering options available to see the issues assigned to you. However, you will need an account to be able to do that.

2.3.4 Socialcast

Think of it as a Twitter or facebook, which only L-mobile employees can access and doesn't have any character limits, allows commenting, adding links, adding images and custom groups. Why are you waiting?

[L-mobile Bluebox Site](#)

2.3.5 Skype

If you haven't heard already, "Skype's text, voice and video make it simple to share experiences with the people that matter to you, wherever they are". Other developers should matter to you. You get the idea.

You can ask your colleagues to find out about the Skype ids of your fellow developers.

To open an account and install Skype please go to the [Skype official page](#).

3 Source Control

In this section, you'll find the source control practices that all developers are required to follow.

3.1 Good to Know

Some remarks about interpreting the rules . * The rules mentioned in the branching section are enforced * Project managers can add a tag to signal last deployed state in a customer branch * *i.e. deployment-lmobile*

3.1.1 General rules of thumb

- Do not be afraid to open branches
- Be afraid when committing to main branches
- Be afraid when merging to main branches
- Do not be afraid to ask

3.2 Concepts

Here are some concepts to get started

3.2.1 Releases

Releases are the safe-to-install points in the development. L-mobile software will be maintained under numbered releases. The numbering system is in the format X.Y.Z where:

- X is the major release number, which indicates a
 - Core change which is backwards incompatible
 - Change-set, which is not offered as a free upgrade to a previous version
 - Milestone, which marks an identifiable completeness of the product
- Y is the minor release number, which indicates the inclusion of an additional, single feature over the previous minor release.
- Z is the build (or revision) number, which may indicate that something broken in the last revision is fixed in this revision. It should be fairly easy to upgrade to a release with only the revision number different.

In the source code management system, releases are tagged as such: release-X.Y.Z but revision can be omitted (release-X.Y) when there is none for the given release.

3.2.2 Release Targets

Release targets are the plans for a major (X) or a minor (Y) version change. They have a due date, and may also have a hard deadline to cover the worst case scenarios. They are declared and discussed in the team meetings. Every change included in the target is added to issue tracker with the proper release field. A release target is finalized in an email message, after the team meetings. Later additions can occur by creating a "Feature Request" issue in the tracker. Unissued features cannot be merged to the default branch.

L-mobile CRM	Release 4
<ul style="list-style-type: none">•Features:<ul style="list-style-type: none">•Feature 1 (Developer A)•Feature 2 (Developer B)•Feature 3 (Unassigned)•Due: DD.MM.YYYY•Deadline: Due + 1 Week	

Release targets

3.3 Branching

Branches are used to separate and organize the changes in the source code.



Branching

Warning Default branch doesn't accept commits other than bug-fixes, merges and merge-fixes since October 2013. All, even the small, features must be developed in a feature branch.

Branches are used to separate and organize the changes in the source code.

All branches, except Stable and Default, have their base major and minor version number in their name. This makes it easier when grafting bug-fixes and merging them to other branches.

3.4 Branches for the main development cycle

These branches are where you go if you are working on the base CRM features.

Warning Default branch doesn't accept commits other than bug-fixes, merges and merge-fixes since October 2013. All, even the small, features must be developed in a feature branch.

3.4.1 Default

Here is where all the feature branches merge. There is a single Default branch.

Warning Default branch doesn't accept commits other than bug-fixes, merges and merge-fixes since October 2013. All, even the small, features must be developed in a feature branch.

Here is where all the feature branches merge. There is a single Default branch.

Important No commits other than bug-fixes and merges are allowed.

- Build: Default branch should always be buildable under the "Debug" and "Release" settings.
- Automated tests: All automated tests need to pass. When broken, they need to go back to working state in 5 days.
- Bugs: Known bugs are not tolerated, and need to be fixed in 5 days. If the bug appears on Stable too, it needs to be fixed there and then merged / grafted back to Default.
- Merging to Default, requires a developer to pledge his/her availability to support and maintain the Default branch for 5 days.

3.4.2 Stable

The branch which is the base for new deployments. There is a single Stable branch.

Important No commits other than tag-additions, bug-fixes and merges are allowed.

- Build: Any point of revision in Stable needs to be buildable under the "Debug" and "Release" settings.
- Automated tests: All automated tests need to pass. **Committing to Stable is strictly not allowed with a broken build or unit tests.**
- Bugs: Known bugs are not tolerated. Any newly discovered bugs, if present, must be fixed here. (Other than those marked with "Won't Fix" in the issue tracker).
- Merging to Stable, requires a developer to pledge his/her availability to support and maintain the Stable branch for 5 days.

3.4.3 Feature Branches

The branches where new features are developed.

Warning Feature branches have their target version in their name, instead of the version of their parent. This is an exception.

The branches where new features are developed. They ultimately merge to the Default branch and also may merge to other feature branches (including customer feature branches) but nothing else. **They should be closed as soon as they are merged to the relevant branches.**

- Build: Can be broken.
- Automated tests: Can be not passing. No test can be changed unless the logic changes. New tests can be added.
- Bugs: All bugs are tolerated.
- Branch naming convention: fb-X.Y-(feature_name)

3.5 Branches for the customer development cycle

Somewhat the same as the main development cycle model, adapted for the customers.

3.5.1 Customer Default Branches

Here is where all the customer feature branches merge.

- Identical to the Default, except...
 - Can't be merged to the Default or Stable in any way.
 - Will be merged to the customer's stable, before a deployment.
- Branch naming convention: cb-X.Y-(customer_name)

3.5.2 Customer Stable Branches

The branches that actually got / gets to be deployed to customer production systems

- Identical to the Stable, except...
 - Can't be merged to the Default or Stable in any way.
 - Will be merged to the customer's default after a bugfix.
- Branch naming convention: cb-stable-X.Y-(customer_name)

3.5.3 Customer Feature Branches

The branches that new features are developed for customers, that are deployed to test systems

- Mostly identical to a feature branch, except...
 - If needs to be merged to the Default too (apart from being merged to the customer default), needs to be branched as a Feature Branch and generalized first, before being merged to Default.
 - Only a single plugin-specific portion of the CRM code can be changed. If customer needs more than one plugin, they need to live in separate Customer Feature Branches until they are merged.
- Branch naming convention: cfb-X.Y-(customer_name)-(feature_name)

3.6 Other Branches

Probably you'll rarely work with these

3.6.1 Experimental Branches

The branches where “proof-of-concept” features are developed

- Not regulated in any way – No guarantees of any stability
- Unlikely to merge to Default.
- When it is decided that an experimental will be picked up for a release, it first needs to be converted to a feature branch and comply with all of its rules.
- Branch naming convention: `experimental-X.Y-(feature_name)`
- After one month of inactivity, an experimental branch can be closed by anyone.

3.6.2 Merge Branches

The branches where big feature branches merge together, before merging to the Default.

- Created only when two feature branches are likely to conflict.
- Will merge to Default and be closed as soon as the solution builds in the release config and the tests pass.
- Branch naming convention: `merge-(feature_1_name)-(feature_2_name)`

3.6.3 Bugfix Branches

Created when a lot of small bug-fixes about a specific part of the application will be applied.

- It can be branched off from stable and merged back to stable, and stable only.
- Named as `bugfix-X.Y-(broken_feature_or_bug_name)`
 - example: `bugfix-3.2-order`

3.7 Operations

Explanations of some common operations in Mercurial, along with the cases when to use them and L-mobile specific conventions.

3.7.1 Commit

Make changes and publish them. There is a single rule, when committing:

Important Make it atomic! (I was very close to making this text huge and blinking. It is THAT important.)

Your commits should be easy to summarize, better when only in one sentence.

Good Examples:

- Bugfix(334512): Button for deleting a contact now works.
- Feature(348234): Generic Lists can now be sorted
- Adds missing files to the repository for rendering Generic Lists (introduced at rev. 348234).

Bad Examples:

- It works!!11 (*What works?*)
- Fixed stuff. (*Stuff??*)
- Fixed Generic Lists (*Which part? There must have been an issue for it, right?*)
- Saved the world. (*Which world? Joke aside, please don't do this. By the way, the maintainer of this documentation is exempt from this.*)

3.7.2 Merge

Merge changes in different branches. Merging may seem like the simplest operation but it is also the source of the mythical Merge Errors. Many common scenarios require at least a merge to complete.

Developers need to know and take care of the following when merging:

- When a conflict happens, take your time to review the changes. Fixing bugs later takes more time.
- You can use a different merge tool (other than kdiff, for example winmerge). Choose as you wish.
- Before pushing the merge, make sure that the solution builds in the release mode.
- Make sure again. Clean the bin folder. Rebuild. Click a few buttons maybe? Do tests pass?
- Do not change the failing tests. Change the failing code. If the logic has changed, consult with the test author.
- Don't merge to Stable, Default or any customer's Stable or Default, if you won't be there to fix it if anything goes wrong for 5 days.
 - This is not a hard limit. If you are making a small change, you can use some common sense. Just be responsible.

Helpful resources

- Resolving Conflicts [video](#)
- Documentation on merging [here](#)

3.7.3 Graft

Copying a the change-set from a single commit to another branch. Grafting is copying a the change-set from a single commit to another branch. It can be used on the following cases:

- A bug-fix is done on the customer branch. It is later discovered that the same bug also exists at Stable.
- A core system change made in a feature branch is needed by other feature branches. (Even though for this change itself, there must have been a separate feature branch but we don't live in a perfect world.)
- A bug-fix in the Stable is needed immediately at another branch.
- Porting a feature from a customer-feature-branch to a feature branch

The same checklist used while merging is also valid for grafts. See [merge](#) operation.

3.8 Roles

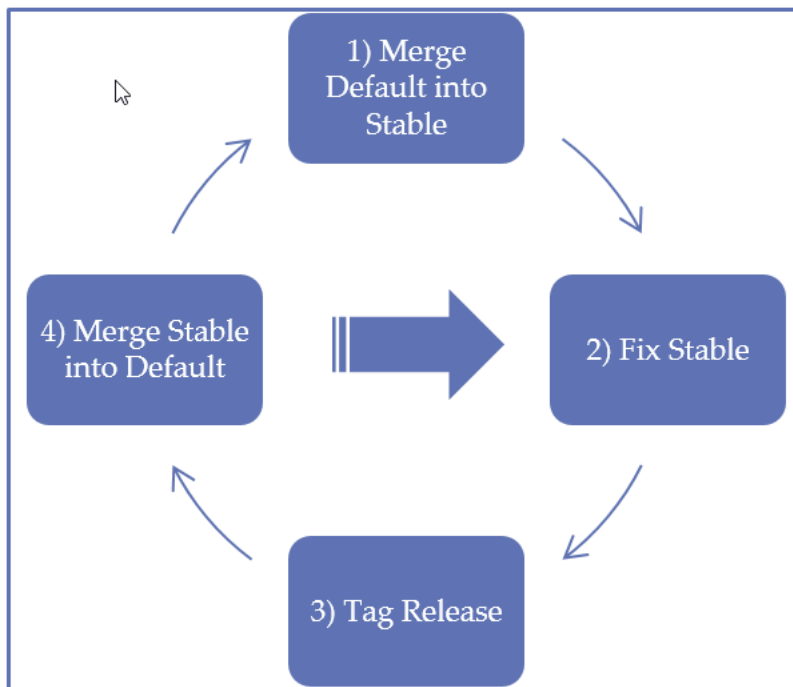
There are different roles for working in the repository.

3.8.1 Maintainers

Maintainers are developers with the special task of making sure that a stable version of the software exists. The usual of a maintainer is as follows:

- Merge the Default into the Stable when either
 - All features in a release are complete
 - A minor or major release needs to be published
- Fix the Stable until it really is stable, when doing so, for each bug, they need to:
 - Fix it
 - Commit change
 - Tag by incrementing the bug-fix number (Z)
 - Merge Stable into Default
- Merge the closed feature branches into the Default

Maintainers are responsible for the stability of the releases. Ensuring stability can be achieved by automatic or manual testing. Maintainers are free to adopt any testing practices to increase efficiency. If and when automatic tests are written and they are passing, developers have to keep them passing. In feature branches, new tests can be added to ensure that the maintainers can easily detect if the additional feature still works.



Maintenance workflow

3.8.2 Useful information for maintainers

Find all closed branches

In tortoise search bar, you can type:

```
closed() and not parents(branch('default')) and keyword('fb-')
```

to get this result

Graph	Rev	Branch	Description	Author	Age	Tags	Phase
4174	fb-dynamic-forms	fb-dynamic-forms	fb-dynamic-forms	chris	1 days		public
4021	fb-3.2-contact-person-for-order	fb-3.2-contact-person-for-order	fb-3.2-contact-person-for-order	age	2 weeks		public
4003	fb-3.2.4-HandTests	fb-3.2.4-HandTests	fb-3.2.4-HandTests	Paula Dem	2 weeks		public
3994	fb-3.2-NativeDropDownPolyfill	fb-3.2-NativeDropDownPolyfill	fb-3.2-NativeDropDownPolyfill	age	2 weeks		public
3920	fb-3.2-value-representative	fb-3.2-value-representative	fb-3.2-value-representative	age	1 weeks		public
3904	fb-project-plugin	fb-project-plugin	fb-project-plugin	Michael Schelle	4 weeks		public
3803	fb-dynamic-notifier	fb-dynamic-notifier	fb-dynamic-notifier	Michael Schelle	4 weeks		public
1010	fb-checklist	fb-checklist	fb-checklist	oliver	5 months		public
509	fb-offline-dispatches	fb-offline-dispatches	fb-offline-dispatches	oliver	12 months		public

Filter for closed branches

3.8.3 Developers

There is nothing to see here yet.

3.8.4 Project Managers

There is nothing to see here yet.

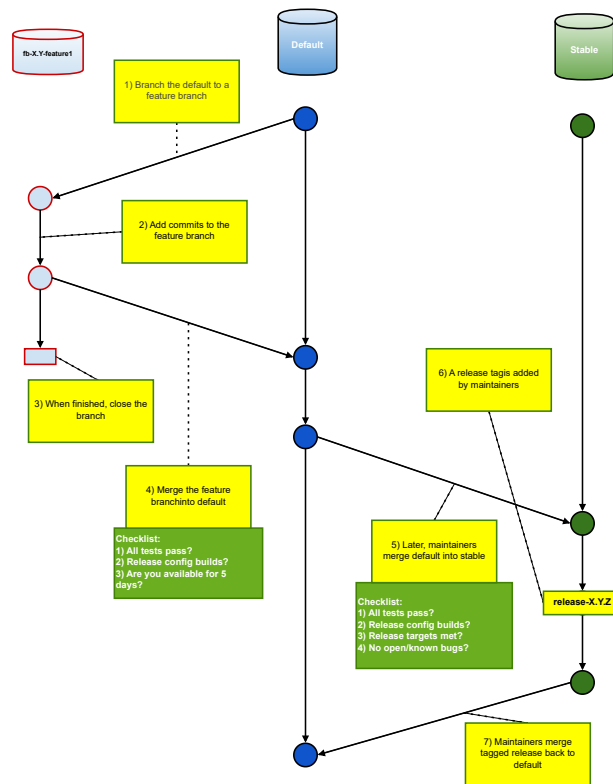
3.9 Common Scenarios

Here are some common scenarios developers face when using source control and ways to deal with those.

Note The svg code in these documents can be edited easily by using software such as <https://draw.io>.

3.9.1 Developing a feature

You can follow these steps when developing a simple feature:



Developing a feature

3.9.2 FAQ

What if it takes too long for my feature branch to be available in the stable? My customer can't wait!

You can merge the feature branch directly to a Customer Feature Branch. In this case, all the burden of testing and maintaining the feature for the customer will be on you. As merging keeps history information of the individual changes, when stable is ready, there will be no problems for merging it in the future to get additional changes.

3.9.3 Development for Customers

4 Concepts of development

This section tries to focus on the bird view perspective of development. Some topics with a broad reach are discussed on a very high level. See further chapters for more details.

4.1 E-Mail Dropbox

E-Mail Dropbox functionality comes in 2 different modes. POP3 and SMTP where each one has different requirements for setup, operation and maintenance.

The general idea behind the E-Mail Dropbox was that users are able to forward or pass message copies to the Crm/Service system with no more additional effort than Forwarding or Blind Carbon Copy a mail to the system.

The system should then inspect the message and find appropriate information to link the message to the corresponding entities. This chapter will introduce you to the technical foundations of the delivery. The inspection of messages is described elsewhere.

If the server is not able to relate the message to any object in the system the original sender receives a so called *bump message*. That message describes the problems the automatic inspection process found with the specific mail so the user can correct the issues before sending new mails.

- Make sure the Crm/Service server has well configured SMTP settings and is able to send mail, otherwise the bump-messages are dropped.

4.1.1 SMTP Dropbox

The SMTP Dropbox is a small embedded SMTP server running inside the Crm/Service instance. It requires the application to be up and running as the SMTP Dropbox is started together with the application.

An SMTP Server is basically a small daemon listening on a specified port for incoming requests. SMTP uses a set of well [documented and simple commands](#) to retrieve the messages and respond to the calling process.

The most common scenario is a company using the Crm/Service system and using Microsoft Exchange as a mail server system. In that case sending mails to the Crm/Server system is as easy as the following steps:

Requirements for the SMTP Dropbox

- Make sure the server is able to send mail (system.net smtp settings)
- Create a subdomain to run the Crm/Service mail domain for e.g. lmobile.example.com
- Add this subdomain to the web.config of the system so users will see their personal addresses ending with this hostname e.g. ab67b8d78@lmobile.example.com
- Create a new Send connector in outlook for the domain mail.example.com
- Specify a Smart Host and port for the Send connector to use
 - Use the IP Address or hostname of your L-mobile Crm/Service server
 - The port that the SMTP E-Mail Dropbox listener runs at can be specified in the jobs.xml file of the Crm/Service application. *Default port is 25*
- Make sure the Firewall on the server allows incoming connections on the specified ports
- Make sure the application is **always-on**. Because the SMTP Server is only started when the Application pool is
 - No Idle timeout
 - Automatic start of the application pool
 - HttpPing if necessary (after periodic recycles)
- After creating the Send Connector all mails sent via your Exchange system will be relayed to the Crm/Service system and processed accordingly
- If you want to use the SMTP E-Mail Dropbox make sure to operate both servers on different ports (e.g. 25 and 26, because each port can only be bound once)

To make the SMTP Dropbox work globally (apart from mails sent using the corporate Exchange server) you would need to make sure all mail servers forward outgoing messages to your Crm/Service SMTP Server. This can be achieved by adding a global [MX DNS record](#) for your subdomain. That way the server must be accessible from the Internet and even anonymous web servers are able to relay to the system.

4.1.2 POP3 Dropbox

The POP3 Dropbox is an earlier implementation of the E-Mail Dropbox. It consists of the idea that you can specify a wildcard E-Mail recipient for each (Sub-) Domain. That way you told your Host that all incoming messages for *@lmobile.example.com would be stored in the central E-Mail mailbox for e.g. dropbox@l-mobile.example.com (think of this as the central postmaster).

The POP3 E-Mail Dropbox Agent is a daemon process running inside the Crm/Service application. It will fire every once in a while (default every 3 seconds) and log in to the central POP3 mailbox checking for new incoming messages. The messages are downloaded from the server and processed afterwards. In any case the messages will be deleted from the mailbox after succesful retrieval.

- Make sure the server is able to send mail (system.net smtp settings)
- You need to specify the subdomain in the web.config of the application
- Prepare the wildcard email mailbox and handling with your domain hosted
- Note the credentials used to login that central mailbox
- Enter the user credentials to the job-data-map of the DropboxAgent in jobs.xml
- Please make sure to use different subdomains and mailboxes for Test and Production to prevent accidental mail download
- Check the mail server and polling settings, sometimes polling a mailbox too often is considered harmful and the mailbox will be locked automatically (to prevent malicious attacks)
- Make sure the mailbox password does not expire. The Dropbox agent is a technical process and no one is going to enter a new password

The SMTP Dropbox should be used wherever appropriate because it will fire only when an E-Mail arrives. You should make sure only one of the 2 Agents is started.

Preface Every developer uses a dedicated instance of the database All changes are recorded in the version control

The diagram illustrates the L-mobile Development Server architecture, divided into three main functional areas:

- Development Environment:** This area includes three developer icons, each with a computer and database icon, representing the "Development + Unit Testing" phase. Arrows indicate the flow of code from the developers to the Subversion Repository.
- L-mobile Development Server:** This central server area contains:
 - Subversion Repository:** Receives code from the development environment and interacts with the CruiseControl.NET Buildserver.
 - CruiseControl.NET Buildserver:** Monitors the repository and triggers builds. It has a "Drop Location" for artifacts.
 - Drop Location:** Stores build artifacts. Dashed arrows point to specific build outputs, such as "Build Number = 1.0.0.124", "1.0.0.123", "1.0.0.122", and "1.0.0.121".
- Installation Process:** This area details the steps to deploy the application:
 - 1 Check out:** Represented by a green arrow pointing down to a document icon.
 - 2 Deploy Baseline (optional):** Represented by a terminal window icon.
 - 3 Apply Migration Scripts:** Represented by a document with a checkmark icon.
 - 4 Database ready:** Represented by a database cylinder icon with a green checkmark.

Workflow Arrows:

- From **Development Environment** to **Subversion Repository**.
- From **Subversion Repository** to **CruiseControl.NET Buildserver**.
- From **CruiseControl.NET Buildserver** to **Drop Location**.
- From **Drop Location** to the **Installation Process** (specifically to step 1).
- A feedback loop from the **Installation Process** back to the **Development Environment** is labeled "Run after every Update".
- A feedback loop from the **Drop Location** back to the **Development Environment** is labeled "Run after Check-Out in Drop Location".

The database can be created using either

- the DeployBaseline.bat script
- downloading an empty database from the TeamCity build artifacts
- automatically as part of the database migrations running on application startup

4.2.2 Test data

The test data is created inside a [OneTimeSetup](#) method `CreateTestData`. It will submit all objects defined under the `TestData` subfolder to the oData API. This requires the `TestData` to be submitted in an order which doesn't violate any foreign key constraints.

4.2.3 Modify aka. Migrations

After setting the Baseline no changes should be made to the create scripts. Instead we need a mechanism to populate changes to existing databases without recreating them every time. I decided to go with an approach that is inspired by ruby migrations. Basically we create objects with a version information attached to change our database.

```
[Migration(20090311134435)]
public class Address_AddSomeColumn : Migration
{
    public override void Up()
    {
        Database.AddColumn("Address", "SomeColumn", DbType.VarChar, 50)
    }
}
```

The Up methods are executed by the framework when applying the migration. After executing the Migration a new version information is stored in the database. We have the power to create new tables, change existing columns and do some more stuff necessary to upgrade the database. The Version is basically a timestamp that makes sure we can upgrade the database until a specific version (date).

4.2.4 Buildserver Integration

The build server uses the same method, getting the latest version of the source code and applying all migrations to bring the database to the current version.

4.2.5 References

[Get Your Database Under Version Control](#) [Versioning Databases](#) [Migrator.Net](#) [Guide to Ruby on Rails Migrations](#)

4.3 Logging

We use Logging very extensively. On the server log4net is used, on the client js logging is available.

4.3.1 log4net - SmtpAppender

- You can use `SmtpAppender` for debugging purposes to send logging messages to your email address.
- Open log4net **config file** in `\src\Crm.WebApp_Data\config\log4net.config`
- **Register** `SmtpAppender` in the root element.
- Enter your **email address** to which the messages should be sent.
- You can let all other configuration as they are - even the dummy email provider. It just works!
- See as well [Apache log4net](#)

Example

```
<log4net>
  <root>
    <level value="INFO" />
    <appender-ref ref="SmtpAppender"/>
  </root>

  <appender name="SmtpAppender" type="log4net.Appender.SmtpAppender">
    <to value="<recipient@example.com>" />
    <from value="<sender@example.com>" />
    <Username value="<sender-user>" />
    <password value="<sender-pass>" />
    <authentication value="Basic"/>
    <subject value="test logging message from log4net" />
    <smtpHost value="<smtp.example.com>" />
    <bufferSize value="512" />
    <lossy value="true" />
    <filter type="log4net.Filter.LevelRangeFilter">
```

```

    <levelMin value="ERROR" />
    <levelMax value="FATAL" />
  </filter>
  <evaluator type="log4net.Core.LevelEvaluator">
    <threshold value="ERROR"/>
  </evaluator>
  <layout type="log4net.Layout.PatternLayout">
    <conversionPattern value="%newline%date [%thread] %-5level %logger [%property{NDC}] - %message%newline" />
  </layout>
</appender>
<log4net>

```

Variables Please remember to fill in the < > marked fields in the above config.

4.4 Testing - Towards reliable state

Tests provide guarantees about the working state of the features you develop.

4.4.1 Selenium Tests

Selenium Tests are user interface tests that check if the user-facing part of the application functions without any errors. A user can programatically click buttons, navigate pages, submit forms and nearly anything else that a user of the application can achieve by using their keyboard and mouse.

Additional Information is available in the [Selenium Documentation](#).

4.4.2 Things to Know Before Writing Tests

There are some important points that you'll need to know before writing Selenium Tests, in order to make them work successfully. Here is the list of them:

- All animations are disabled when pages are called with a token
- Try to prevent login whenever possible, directly call the url you want to test and do it there instead of coming the long way home
- A token is automatically added from the main call to all subsequent ajax requests and forms
GoToUrl(url , token)
[[warning: There still isn't way to append tokens on redirects]]
- There are a bunch of WebDriverExtension methods to make sure you can wait for almost every situation
- Always add a bunch of extra time-out seconds as the server needs them
- Take care when checking the Visibility in Selenium WebElements (problematic with validation spans)
- Always create a page object to prevent Selectors flying all over the place
- Refactor your tests whenever they feel cluttered
- Think of using CSS Selectors instead of XPath as they tend to be a little faster

4.4.3 Unit Test Data

Whenever you're writing Selenium Integration tests it is pleasant to have some Test-Data available which gets populated before running the Integration test suite. To achieve this a set of Data is created prior to running the unit tests. The test data is scripted in the *.sql files in the folder src/Crm.Database/Create/TestData. It should be published right after running the DeployBaseline and before running any migrations. The test data should contain only relevant data, this means it will populate real or fictive Companies instead of dummy data like "A new company" etc.

One goal is to have a good set of test data available that is recreated reliably and doesn't change between releases. This way a demo system for sales purposes can be created from the command line.

4.5 Translations

It is very important to add a translation to every non-user-generated string, as the CRM can be used with a growing list of languages.

4.5.1 How to Add a Translation

[[thumbnail:translation.png]] Adding a translation is as easy as opening the corresponding resource file for the language you want to edit and adding a new line with a unique key that represents the string you want to add and the value, which will be used as the translation when the application is viewed in the language you are editing. Adding a comment, and specifying where that string is used (even though the key should clearly indicate that as well) also helps other people who may re-use the translation a lot, while preventing any misuse.

4.5.2 How to Use a Translation

A translated string is easily inserted to a page by calling a function as the following:

```
<%= Html.Localize("TheTranslationKey", "Fallback text (Displayed if no translation present)") %>
```

4.5.3 Things Not to Forget

- Every translated string **MUST** be a sentence or a word with a meaning **on its own**. Translations of single words like "of", "a" and "so" aren't possible for many languages without a context.
- Comments and self explanatory keys are encouraged and very important for re-usability.
- Translation keys must be static. User generated content or anything that is read dynamically **MUST NOT** be translated.
- Translations **MUST NOT** be parts of sentences.
- Translations **CANNOT** have embedded HTML.

4.6 Targeting both Offline and Online modes

This page will include some information that may be handy when developing components which work both in online and offline modes for L-mobile Crm.

5 Integrating ERP systems (Our Legacy Integration story)

Almost all Crm or Service installations will have to share data with the customers ERP or other legacy systems. The integration of data will allow for process automation between the systems. Existing data can be reused and does not have to be entered multiple times to different databases.

The first step to integration is almost always read only integration. This means data is transferred from the ERP system to the Crm/Service system to fuel the Sales or service operations. In a second step data could be transferred from user input or process output of the Crm / Service system to further processing in the legacy (ERP or other) system. When it comes to write integration the integrity and quality of the to be transferred data will need to satisfy the legacy systems demand.

This document will explain the aspects of legacy integration. It will cover our preferred integration strategy and give insight to some viable alternative routes if that becomes necessary during the project.

5.1 Architectural considerations

Most legacy systems use a database of one or the other kind. Often to be found systems include Oracle, MS SQL Server, Progress, DB/2 or other kinds of legacy or vendor specific storage solutions. The key aspect here is that the data is stored in a central, well structured and machine readable place. For an integration it comes down to the 3 main questions:

- How can one connect to the storage system
- What is the structure of the data (is there documentation or a person that could help with the integration)
- How do the servers interact on a network level basis (different subnets, firewall restrictions, port forwarding, etc.)

Once these questions are answered one can start with the basic integration of the legacy system. But keep in mind that for a production ready integration you need to take into account aspects like: transparency, performance, logging and stability. We will show some aspects of creating a successful integration in the later chapters of this document.

5.2 Integrating database systems

As you probably know the main storage for the Crm / Service system is a SQL Server database that will be hosted on a SQL Server instance. Your task is to grab the relevant data structures from the legacy system and transfer them to the Crm / server structures. The easiest way to achieve this kind of integration is mapping the remote storage of the legacy system to a normalized form inside of the SQL Server instance of the Crm / Service database. Fortunately SQL Server is able to connect to a bunch of database systems using the so called "Linked server" technology.

A linked server acts like a proxy to the remote database system. You can issue queries right from your SQL Server Management studio. But keep in mind, that when using linked servers you should use the generic version of T-SQL that is implemented throughout the different databases. You can always opt to implement queries in the native dialect of the linked server using tools like OPENQUERY.

Connecting to another database instance often requires specification of remote credentials for authentication. The credentials only need read only permissions, that should be no problem to obtain from the administrator of the project.

5.2.1 MS SQL Server (MS Dynamics NAV, MS Dynamics AX)

Connecting to another instance of SQL Server is fairly straightforward. You can find instructions to create a new linked server here.

5.2.2 Oracle (Infor Erp Com, Infor Erp Blending, etc.)

Connecting to an Oracle database requires a little more of preparation. SQL Server does not natively contain drivers to connect to the remote instance. But there are ways to add a linked server using a combination of Oracle software. Please keep in mind that you need the right version of tools for your host database instance. Most SQL Server installations are 64bit nowadays. So please take care to use the right packages.

5.2.3 Progress (ProAlpha)

The last known good way to connect to a Progress system and transfer data includes creation of an System DNS ODBC connection using the Progress ODBC driver. Adding a linked server to the System DNS then allows you to access the remote database.

5.2.4 Other database systems / CSV file integration

A ODBC driver exists for most of today's database systems as this is the de-facto standard for database integration. If your system was not present in the former list please assure that the vendor of the ERP database system supports standard ODBC access to the database. Create a System-DNS entry and access the database from the Linked Server for ODBC function in MS SQL Server.

Other forms of integration allow to read data from CSV or other structured files (XML, etc.). But using this kind of files the risk of malformed files increases.

5.2.5 Programmatic import

For some ERP systems (like SAP) no direct database access will be granted. That means the solutions described in this document won't apply to your project. Under all circumstances you will want to try to evaluate database access nonetheless due to the improved performance and maintainability. If no other options exist you will probably find yourself writing a set of C# files that execute the import by reading from Webservices, Remote APIs or other locations.

5.3 The integration folder

When developing a legacy system integration it is recommended to stick to a set of best practices. This enables other developers to help out and understand the structure quickly. In the later chapters references are made to some templates that can be used to create new or extended integrations.

Common Integration structure pattern

```
src
  Crm.Web
    Plugins
      Customer.XXX
        Integration
          Import
            I_External_X.sql
            I_External_Y.sql
          Linked Server
          Views
            V_External_X.sql
            V_External_Y.sql
          IMPORT_Daily_Prod.bat
          IMPORT_Daily_Test.bat
          IMPORT_Periodic_Prod.bat
          IMPORT_Periodic_Test.bat
```

The integration itself is not part of the actively maintained standard product at the creation of the document (April 2015)

5.4 Normalizing legacy data to known formats - the X_External story

Once the connection to the remote database system is established one can start to create SELECT statements against the remote database structure. But this causes new challenges. Most of the time the database will have weird field and table names. These should not leak from the remote database structure to the Crm / Service system for 2 reasons:

- The mapping between remote and local storage structures should be done at a central and maintainable location
- All scripts that interact with the remote structure need to know the specific implementation leaving less room for reuse of once existing scripts

To solve both building an interface to the legacy system involves creation of normalized structures referred to as V_External (Views) or T_External (Physical tables). Depending on the abilities of the remote database system choose one or the other. This document will focus on using external views for normalization.

It is preferable to create an individual view for each entity that will be integrated with the legacy system. The views will always have a similar structure:

```
CREATE VIEW V_External_Company
AS
SELECT
  a.FieldWithCompanyNo AS CompanyNo
  ,a.FieldWithCompanyName AS Name
  ,BINARY_CHECKSUM(
    a.FieldWithCompanyNo
    ,a.FieldWithCompanyName
  ) AS LegacyVersion
FROM LinkedServer.Schema.RemoteTable a
```

or for Oracle using OPENQUERY

```

CREATE VIEW V_External_Company
AS
SELECT a.*
      ,BINARY_CHECKSUM(
        a.CompanyNo
        ,a.Name
      ) AS LegacyVersion
FROM OPENQUERY(LINKEDSERVERNAME,
  'SELECT
    FieldWithCompanyNo AS CompanyNo
    ,FieldWithCompanyName AS Name
  FROM RemoteTable') AS a

```

Writing views like this focusses on the main aspects of mapping, the remote data selection process and converting the Remote structure to a local normalized structure wherever necessary. When type conversion between remote and local structure becomes necessary this should go into the views as well to make sure it is maintained at a central location.

For the creation of the views one should focus to keep the view files outside of the database so these can be easily transferred between the development and customer environment.

5.5 Importing from your normalized tables - Merge

When it comes to importing from the normalized structures one SQL Server tool is the key to keeping things nice and clean. Meet the [SQL Server MERGE statement](#). The SQL Server merge statement basically allows to use a set of source records and merge them to your given target destination based on user defined conditions. This allows us to read data from the normalized structure and conditionally insert or update records in the target Crm / Service structure.

Merge statements consists of some common elements:

- Merge source
- Merge target
- Link condition e.g. What is the primary link between source and target
- What to do if target does not contain a record aka WHEN NOT MATCHED
- What to do if record exists in source and target aka WHEN MATCHED
- What to if record stops to exist in source aka WHEN NOT MATCHED IN SOURCE

Plus there is a nice addon called OUTPUT clause. The output clause gives you the results of the MERGE operation in a separate table. This table can be inspected and reused for other operations. Using the keyword *\$action* you will be able to distinguish INSERT, UPDATE and DELETE operations.

5.6 The two step import

In the Crm / Service system many tables exists that contain of records in at least 2 tables. The inheritance of classes forces the common elements to be stored in a root table that contains some kind of Discriminator ([read about it here](#)). Most of the time the specific attributes for a class are then stored in a second table using the primary key of the root table as foreign key.

For integration of legacy systems this creates a new challenge: You have to merge to the root table, grab the inserted primary keys and store them temporarily to merge a second time to the extended table using the primary key from the root as a foreign key.

But fear not, there are common scripts that will help you understand and achieve the former.

5.6.1 Prepare input storage

First we prepare an intermediate storage for our 2 step import process. This sounds odd at first, but you will find out about the use of this table later:

```

IF OBJECT_ID('tempdb..#ContactImport') IS NOT NULL DROP TABLE #ContactImport
CREATE TABLE #ContactImport (Change NVARCHAR(100),
                             ContactId INT,
                             LegacyId NVARCHAR(100))

```

Now we prepare our temporary import storage by reading from the source structure into a temporary storage table

```

IF OBJECT_ID('tempdb..#Import_Company') IS NOT NULL DROP TABLE #Import_Company

SELECT
  v.CompanyNo
  ,v.Name
  ,v.LegacyVersion

```

```

INTO #Import_Company
FROM V_External_Company AS v

```

Please note the INTO #Import_RemoteTable part of the select statement. This way you don't need to declare the full table structure, using this and your view declaration the implicitly created table structure for #Import_RemoteTable will match the desired outcome.

5.6.2 Create indices in temporary table -> Performance boost

Creating an index in your temporary import table will speed up the merge operation dramatically.

```

CREATE NONCLUSTERED INDEX IX_#Company_CompanyNo ON #Import_Company ([CompanyNo] ASC)

```

5.6.3 Merge operation - Step 1

In the first step we will take all data from the temporary import storage that does not exist in CRM.Contact table and insert records accordingly. Please note the conditions in the WHEN MATCHED part. The checks on LegacyVersion prevent unnecessary updates to the table (reducing table locks and performance problems).

```

MERGE [CRM].[Contact] AS [target]
USING #Import_Company AS [source]
ON [target].[LegacyId] = source.[CompanyNo]
AND [target].[ContactType] = 'Company'
-- We already know a record with this LegacyId
WHEN MATCHED
    AND ([target].[LegacyVersion] IS NULL OR [target].[LegacyVersion] <> [source].[LegacyVersion])
    THEN
        UPDATE SET
            [target].[LegacyId] = source.[CompanyNo]
            ,[target].[LegacyVersion] = source.[LegacyVersion]
            ,[target].[ModifyDate] = getutcdate()
            ,[target].[ModifyUser] = 'Import'
            ,[target].[Name] = source.[Name]

-- If not found we try to insert with the appropriate data
WHEN NOT MATCHED
    THEN
        INSERT
        (
            [ContactType]
            ,[LegacyId]
            ,[LegacyVersion]
            ,[IsExported]
            ,[Name]
            ,[IsActive]
            ,[Visibility]
            ,[CreateDate]
            ,[ModifyDate]
            ,[CreateUser]
            ,[ModifyUser]
        )
        VALUES
        (
            'Company'
            ,source.[CompanyNo]
            ,source.[LegacyVersion]
            ,1
            ,source.[Name]
            ,source.[IsActive]
            ,4
            ,getutcdate()
            ,getutcdate()
            ,'Import'
            ,'Import'
        )

-- Record was previously imported and is still active in target
WHEN NOT MATCHED BY SOURCE
    AND [target].[IsActive] = 1 AND [target].[ContactType] = 'Company' AND [target].[LegacyId] IS NOT NULL
    THEN
        UPDATE SET
            [IsActive] = 0
            ,[ModifyDate] = GETUTCDATE()
            ,[LegacyVersion] = NULL

```

```
-- All records to the temp table including their action
OUTPUT $action
      ,inserted.ContactId
      ,[source].CompanyNo AS LegacyId
INTO #ContactImport;
```

With this statement the first part of the tables get published. But once you have a record in CRM.Contact you want to make sure to add to the CRM.Company table as well to make things complete. Please note the OUTPUT clause in the statement which makes sure all merge results get transferred to your intermediate storage table. This table will contain only the most necessary informations to complete the

5.6.4 Merge operation - Step 2

For the 2 step merge process we will take the results of the first merge step into the root table, join them with the original source table and import to the extended entity storage table.

```
MERGE [CRM].[Company] AS [target]
USING (SELECT ci.ContactId, c.*
      FROM #Import_Company c
      JOIN #ContactImport ci ON c.CompanyNo = ci.LegacyId) AS [source]
ON [target].[ContactKey] = [source].[ContactId]
-- For all new records we insert
WHEN NOT MATCHED THEN
  INSERT
  (
    [ContactKey]
    ,[ShortText]
    ,[SearchText]
    ,[CompanyTypeKey]
    ,[IsOwnCompany]
  )
VALUES
  (
    source.[ContactId]
    ,source.[Name]
    ,source.[Name]
    ,'Customer'
    ,0
  )
-- For all found temp records we update accordingly
WHEN MATCHED
  THEN
  UPDATE SET
    [ShortText] = source.[Name]
    ,[SearchText] = source.[Name];
```

Here you see considerably less complexity. By joining the results of the first merge step and the original source

5.7 Importing data for distributed usage, adding offline clients to the mix

When it comes to adding offline clients to a system, one of the most important things to consider is the legacy data integration. Offline clients will sync their local storage depending upon the modification date found in the source tables of the remote Crm / Service database. This is due to the fact that the clients only want to sync new or changed records to keep network traffic at a low rate.

Therefore it is extremely important to make sure you update your ModifyDate columns during update operations using UTC time stamps. On the other hand you will only want to update the records if this becomes necessary. Calculating a record hash over the imported legacy records helps you make that decision.

5.8 How to import in a transparent fashion, logging and surveillance

When it comes to the maintenance of Import structures we found it very helpful if the Import scripts would create Log files or log messages in the CRM.Log table. For this purpose one can sprinkle some debugging statements into the merge process to help understand what's going on.

5.8.1 Original row count

To know about the amount of records being used in the temporary import storage in the first place.

```
DECLARE @logmessage NVARCHAR(4000);
DECLARE @count bigint;

SELECT
    ...
INTO #Import_Company
FROM V_External_Company

SELECT @count = COUNT(*) FROM #Import_Company
SELECT @logmessage = CONVERT(nvarchar, @count) + ' Records transferred to input table'
PRINT @logmessage
```

5.8.2 Intermediate records

To find out about the amount of records transferred into the intermediate storage. Please take a close look at the Update count as it will most times relate to the amount of records being updated on the mobile client after a sync.

```
MERGE
    ...
OUTPUT $action
    ...
INTO #ContactImport;

SELECT @count = COUNT(*) FROM #ContactImport
SELECT @logmessage = CONVERT(nvarchar, @count) + ' records processed to intermediate table'
PRINT @logmessage

SELECT @count = COUNT(*) FROM #ContactImport WHERE Change = 'INSERT'
SELECT @logmessage = CONVERT(nvarchar, @count) + ' records inserted'
PRINT @logmessage

SELECT @count = COUNT(*) FROM #ContactImport WHERE Change = 'UPDATE'
SELECT @logmessage = CONVERT(nvarchar, @count) + ' records updated'
PRINT @logmessage
```

5.8.3 Try Catch

When your import script has an error at one place or the other you probably want to log the exception to the CRM.Log structures. You can wrap all parts of the import process in a Try ... Catch block allowing for further inspection of the error.

```
BEGIN TRY
    ... Prepare intermediate table
    ... Prepare input table
    ... Merge 1st step
    ... Merge 2nd step
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT
        @ErrorMessage = ERROR_MESSAGE(),
        @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE();

    PRINT @ErrorMessage
END CATCH
```

5.9 Triggering import scripts

As we keep the import scripts as separate sql files in the integration folder it is easy to trigger the import using the command line tool `osql` which is provided as part of the SQL Server tools installation. You will want to create a bunch of batch files expressing what system and interval you're going to import to e.g. `IMPORT_DAILY_Prod.bat`

The file will contain the list of to be executed scripts together with some paths for logging:

```
set db-server=SQL Server
set db-name=LmobileProd
set log-folder=C:\Integration\Log
set integration-folder=C:\Integration

osql -E -S %db-server% -d %db-name% -i "%integration-folder%\Import\I_External_CompanyType.sql" -o "%log-folder%
...
```

This batch file will then be scheduled using [windows scheduled tasks](#).

5.10 Common Pitfalls

As with all complex systems adding database integration to a project can cause a lot of potential for errors. In some circumstances you will want to add additional functions to your import scripts or more complex logic to decide if a record needs to be updated or not.

- Try to question the import of the source data. Try to keep integrated row count as small as possible.
- When updating your data with external sources make sure to have a conflict resolve strategy (data loss can occur)
- Remember to store LegacyVersion to prevent unnecessary updates
- Always remember that Create and ModifyDate shall be stored using UTC instead of local time
- Try to keep import scripts as brief as possible, prevent additional code for better readability / maintenance
- BINARY_CHECKSUMs are okay but have their limitations: Example if a Decimal number changes by a factor of 10, e.g. 100,00 to 10,00 you will get the same checksum. -> Convert them to string before calculating the checksum
- Use a database connector for the legacy database that is compatible with your host architecture (e.g. 32 or 64bit)
- Try to store data from remote location in temporary host tables to prevent unnecessary cost intensive reads
- Use local temporary tables (#table) instead of table variables (@table) to optimize memory usage with big tables
- Don't join remote and local tables unless it becomes really necessary, joining hits performance a lot

6 Development in the customer plugin

6.1 Create a new customer plugin

- Content
- Controllers
- Model
- Views

If necessary, create additional folders and sub-folders. The following folders are of particular importance:

- Database Database This folder can contain two sub-folders:
- Create Contains the SQL scripts (e.g. for creating tables) that are only executed when the plug-in is activated for the first time.
- Migrate Contains the DB migrations. The migrations are carried out each time the plugin is activated. Strictly speaking, the name of this folder can be freely selected, since the migrations are only searched for and found using the migrate attribute.

When the plugin is activated for the first time, a table `dbo_SchemaInfo_` is created. The create scripts in the Create folder are only executed if this table does not exist. The scripts are executed in the order of their alphanumeric sorting, i.e. `1_Skript1.sql` before `2_Skript2.sql` etc.

- add web.config
- Open project properties (right-click on project, select Properties in the context menu). Click the Web tab. Under Servers select "Use Custom Web Server" with URL `http://crm.l-mobile.com`. If you leave the setting at the default value "Use Visual Studio Development Server", an instance of the development server is started for each plug-in project.
- Create a plug-in class in the root directory, which is derived from the Plugin class in the `CrmLibrary.Modularization` namespace in `CrmLibrary.dll`.
- Provide this class with the plugin attribute from the same namespace.

A parameter "Requires" can be transferred to the attribute, which indicates the dependencies of the current plug-in on other plug-ins in the form of a comma-separated list.

```
[Plugin(Requires="Crm.MyFirstPlugin,Crm.MySecondPlugin")]
public class MyThirdPlugin : Plugin
{
}
```

The name of the class can be freely chosen. The dependencies on other plugins are specified by specifying the project name or the name of the .dll file (without the .dll extension). Both are identical by default, but the name of the generated .dll file can be changed in the project properties.

6.2 Routing

To register the plugin-specific routes in the root directory, create a class which is derived from `RouteRegistrar` and implements the `IRouteRegistrar` interface. Provide this class with the `RouteRegistrar` attribute (all from the `CrmLibrary.Modularization.Registrars` namespace). A `Priority` parameter can be passed to the attribute, which specifies the order in which the routes are registered. The order of registration is in descending order of priority, ie it would theoretically be possible for one plugin to overwrite the route of another plugin.

The routes are registered by overwriting the `RegisterRoutes` method of the `RouteRegistrar` base class.

Example

```
[RouteRegistrar(Priority = 10)]
public class VisitReportRouteRegistrar : RouteRegistrar, IRouteRegistrar
{
    endpoints.MapControllerRoute(
        null,
        "Customer.Example/{controller}/{action}/{id?}",
        new { action = "Index", plugin = "Customer.Example" },
        new { plugin = "Customer.Example" }
    );
}
```

The name of the class can be freely chosen. The file name with the class definition is usually called `Routes.cs`, but can also be chosen as desired.

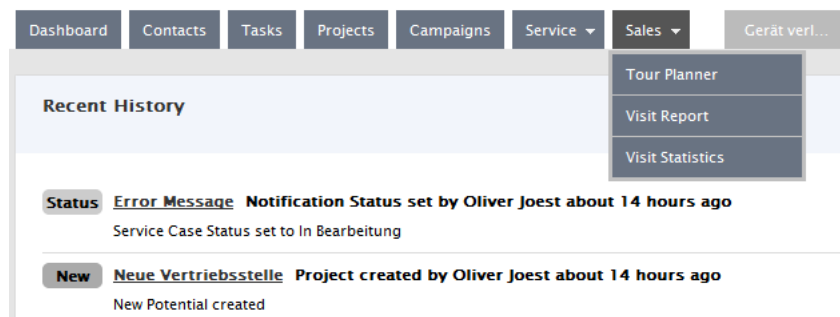
6.3 Menus

If the plugin is to insert menu items, a class is required for each menu item, which MenuRegistrar derives from the namespace CrmLibrary.Modularization and is identified with the menu attribute. Various parameters can be transferred to the menu attribute:

- **Category** Is required if a menu item with submenu items is to be created. The categories of the submenu items must correspond to the title of the upper menu item.
- **Title** The resource key used to translate the menu text.
- **RouteName** The name of the route that is called when the link is clicked.
- **Url** Instead of the route name, an url can optionally be specified. This would be necessary, for example, if the url should contain a query string.
- **Priority** The priority defines the order in which the menu items are displayed.- **HasSubmenus** Indicates whether a menu item has submenu items. At the moment only one level of submenu items is allowed.
- **RequiredPermission** Specifies the permissions a user must have in order to be able to select the menu item. If he does not have the authorization, the menu item is not visible. A comma-separated list of authorizations can be passed to the parameter. This is an OR link, ie the user must have at least one of the authorizations for the menu item to be visible.
- **RequiredRole** Specifies the role that a user must have, similar to the authorization, so that the menu item is visible to him

TODO: Insert Code snippet here

Default



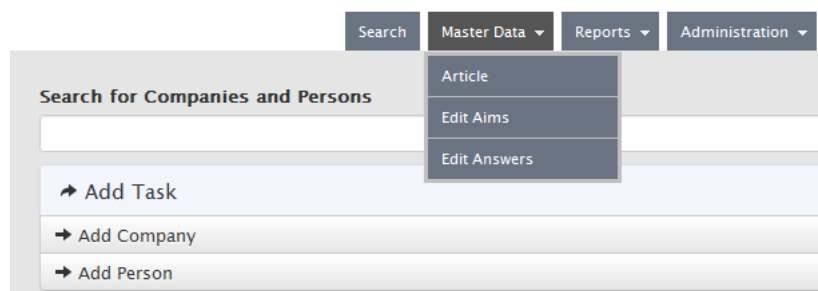
Vom Plugin eingefügter Hauptmenüpunkt mit Untermenüs

In order to add a submenu item to the main menu item "Master Data" or "Master Data", "MasterData" must be specified as the category.

Example

```
[Menu(Category = "MasterData", Title = "EditVisitPossibleAims",
      RouteName = "VisitAims", Priority = 70,
      RequiredPermission = "EditVisitPossibleAims")]
public class VisitAimsSubMenu : MenuRegistrar
{
}

[Menu(Category = "MasterData", Title = "EditVisitReportPossibleAnswers",
      RouteName = "VisitReportPossibleAnswers", Priority = 60,
      RequiredPermission = "EditVisitReportPossibleAnswers")]
public class VisitReportPossibleAnswersSubMenu : MenuRegistrar
{
}
```



Vom Plugin eingefügte Untermenüpunkte im Stammdaten-Menü

6.4 ActionFilter

A plugin can dynamically add `ActionFilter` to action methods, also to existing methods of the main program or other plugins. For this purpose, an implementation of the abstract class `ActionFilterRegistrar` must be written, which must also be identified with the attribute `[ActionFilterRegistrar]`. The abstract methods `RegisterActionFilters` and `GetActionFilterType` are to be implemented, whereby `GetActionFilterType` must return the concrete type of the `ActionFilter` to be registered.

Example

```
[ActionFilterRegistrar]
public class TaskListActionFilterRegistrar : ActionFilterRegistrar
{
    public override IEnumerable<IActionFilterCriteria> RegisterActionFilters()
    {
        var taskListActionFilterCriteria = new ControllerActionCriteria();

        taskListActionFilterCriteria
            .AddMethod<ServiceOrderController>(c => c.Details(null));
        taskListActionFilterCriteria
            .AddMethod<ServiceCaseController>(t => t.Details(default(int)));

        return taskListActionFilterCriteria.AsEnumerable();
    }

    public override Type GetActionFilterType()
    {
        return typeof(TaskListActionFilter);
    }
}
```

6.5 Prevent deletion of lookups

There are cases in which an entry in a lookup table must not be deleted because the lookup value is being used by another entity. Examples are the project status or the project category. If you were to delete a project status or a project category that is used by a project, an exception would occur if an attempt is made to access the lookup property of the project instance.

To prevent used lookups from being deleted, a derivation of the base class `UsedLookupsProviderBase` must be implemented and given the attribute `[UsedLookupsProvider]`. The class implements the abstract method `GetUsedLookupKeys`, which must return a list of the lookup keys used, which must not be deleted.

Example

```
[UsedLookupsProvider]
public class ServiceUsedLookupsProvider : UsedLookupsProviderBase
{
    public override IEnumerable<object> GetUsedLookupKeys(Type lookupType)
    {
        if (lookupType == typeof(Skill))
        {
            return GetInstance<IServiceOrderService>().GetUsedSkills();
        }

        return new List<string>();
    }
}
```

The extension method `ILookup.IsUsed` can then be used in the code to check whether a specific lookup instance can be deleted or not.

Example

```
[HttpPost]
public virtual ActionResult DeleteLookup(ILookup lookup)
{
    if (lookup.IsUsed())
    {
        return Json(new { errorMessage = "LookupDeletionDenied".GetTranslation() });
    }
    ...
}
```

6.6 Events

With the help of events it is possible for a plugin to expand the functionality of the basic program. The extension is only possible at predefined points in which the main program (or another plugin) publishes an event.

The event is an instance of a class derived from `CrmEvent` with any properties that are evaluated by an event handler to be written by the user

Example

```
TODO: Update code example
public class CompanyDeletedEvent : CrmEvent
{
    // Ids of the deleted companies
    public IList<int> CompanyIds { get; protected set; }

    public CompanyDeletedEvent(params int[] companyIds)
    : this(new List<int>(companyIds))
    { }

    public CompanyDeletedEvent(IList<int> companyIds)
    {
        CompanyIds = companyIds;
    }
}
```

This event can be published with the help of the `EventAggregator` class:

```
TODO: Update code example
public void DeleteCompany(int companyId)
{
    var company = companyRepository.Get(companyId);
    if (company == null) return;

    companyRepository.Delete(company);

    CrmEventAggregator.Publish(new CompanyDeletedEventArgs(companyId));
}
```

In order to intercept and process the event, an event handler must be implemented, which expects an object of the event to be processed as the only parameter and must be provided with an `EventHandlerAttribute`:

```
TODO: Update code example
[ConcreteType(typeof(IProjectService))]
public class ProjectService: IProjectService
{
    [EventHandler(typeof(CompanyDeletedEvent), typeof(IProjectService))]
    public void OnCompanyDeleted(CompanyDeletedEvent e)
    {
        var projectsToDelete = from p in projectRepository.GetAll()
                                where p.Parent != null
                                && e.CompanyIds.Contains(p.Parent.Id)
                                select p;

        foreach (var project in projectsToDelete)
        {
            projectRepository.Delete(project);
        }
    }
    ...
}
```

In the example, when a company is deleted in the `EventHandler`, which can be located in a project plug-in, for example, all projects associated with the company are deleted.

The `EventHandler` must be located within a class registered with the IoC container. In the example, the class `ProjectService` is registered as a specific type of the `IProjectService` interface.

The `EventHandler` attribute expects the type of event to be processed as the first parameter and the type of the interface under which the class was registered with the IoC container as the second parameter.

Note: An `EventHandler` for a specific `CrmEvent` class also receives all events derived from it.

Example

The `OnCrmEvent` method of the `NoteService` class receives all events and checks for each received event whether there is an associated implementation of the `NoteGenerator` class. If such a `NoteGenerator` exists, it creates a note about the event (see section `NoteGenerator`).

```
[ConcreteType(typeof(INoteService))]
public class NoteService : INoteService
{
    [EventHandler(typeof(CrmEvent), typeof(INoteService))]
    public void OnCrmEvent(CrmEvent e)
    {
        var noteGenerators = Exports.Container.GetNoteGenerators();
        foreach (INoteGenerator noteGenerator in noteGenerators)
        {
            if (noteGenerator.EventType != e.GetType())
                continue;

            var note = noteGenerator.GenerateNote(e);

            SaveNote(note);
        }
    }
}
```

6.6.1 NoteGenerator

If a note is to be generated for an event, a concrete derivation of the abstract NoteGenerator class must be implemented. The type parameter TCrmEvent corresponds to the event for which the note is generated. The concrete implementation must overwrite the method GenerateNote, which receives the event as an input parameter and uses it to generate and return a Note object.

Example

```
[NoteGenerator(typeof(ProjectCreatedEvent))]
public class ProjectCreatedNoteGenerator : NoteGenerator<ProjectCreatedEvent>
{
    public override Note GenerateNote(ProjectCreatedEvent e)
    {
        var project = e.Project;

        var note = new ProjectCreatedNote
        {
            IsActive = true,
            ContactId = project.Id,
            Text = ProjectStatus.Open.Key
        };

        return note;
    }
}
```

6.7 PluginRenderActions

A plugin can inject its own PartialView into views of the main program or other plugins at predefined extension points. The extension points are to be defined using the PluginRenderActions method, to which the name of the extension point is to be passed as the first parameter.

Example of an extension point

```
<% Html.PluginRenderActions("CompanySidebarExtensions"); %>
```

A method must be implemented within a controller class which returns the view to be injected and which must be marked with the attribute [PluginRenderAction]. The attribute expects a comma-separated list of the names of the extension points at which the view is to be injected as the only parameter.

Example

```
public class CompanyExtensionController : Controller
{
    [RenderAction("CompanySidebarExtensions", "CompanyMobileExtensions")]
    public virtual ActionResult CustomerCard(int id)
    {
        var model = new CrmModelItem<Company>
        {
            Item = companyService.GetCompany(id)
        };
    }
}
```

```
        return View("CustomerCard", model);~~~~~
    }
}
```

▼ Contacts		Add
Beinhauer, Markus Prokurist	+49 4471 966 0 markus.beinhauer@example.com	☆
Appelmann, Tim Leiter IT	+49 4471 966 0 tim.appelmann@example.com	
Balmer, Ralf Leiter Instandhaltung	+49 4471 966 0 ralf.balmer@example.com	☆
Blasebeck, Dieter Supply Chain Manager	+49 4471 966 0 dieter.blasebeck@example.com	

Bereich in der Company-Details-Sidebar in den Plugins einen eigenen View injizieren können

▼

Contacts

Add

Beinhauer, Markus

Prokurist

+49 4471 966 0

markus.beinhauer@example.com

☆

Appelmann, Tim

Leiter IT

+49 4471 966 0

tim.appelmann@example.com

Balmer, Ralf

Leiter Instandhaltung

+49 4471 966 0

ralf.balmer@example.com

☆

Blasebeck, Dieter

Supply Chain Manager

+49 4471 966 0

dieter.blasebeck@example.com

▼

Customer Card

Business Partner No.: –

Region: –

Visit Frequency: **0 per Year**

Indoor Service: –

Term of Payment: –

Kind of Treatment: –

Credit Limit: –

Discount Group: –

Discount 1: –

Discount 2: –

Customer Group: –

Company-Details-Sidebar mit injiziertem View

6.7.1 List of available PluginRenderActions

CrmSite

View	Category
Company/Details.aspx	CompanyHeaderRightInfo
	CompanyTabHeader
	CompanyTab
	CompanySecondaryAction
	CompanyContextExtensions
	CompanySidebarExtensions
Company/Edit.aspx	CompanyEditSidebarExtensions
Contact/BulkControls.ascx	ContactSearchBulkActions
Contact/BulkInfoArea.ascx	ContactSearchBulkInfoArea
Contact/ListActions.ascx	ContactSidebarContextExtensions
Dashboard/Index.aspx	DashboardSidebar
Person/Details.aspx	PersonSecondaryActions
	PersonContextExtensions
	PersonSidebarExtensions
Project/Details.aspx	ProjectExtensions
Shared/SourceSelectBox.ascx	ContactSource
User/Details.aspx	UserDetailsMapSidebarExtensions

Crm.Service

View	Category
ServiceOrderMaterial/ServiceOrderMaterialEdit.ascx	ServiceOrderMaterialEditorExtensions
ServiceOrderTime/ServiceTimeEdit.ascx	ServiceOrderTimeEditorExtensions
Shared/ServiceOrderMaterialNew.ascx	ServiceOrderMaterialEditorExtensions
Shared/ServiceOrderTimeNew.ascx	ServiceOrderTimeEditorExtensions

6.8 Extension Properties

With the help of extension properties, a plugin can extend the edit view of an entity. For this purpose, EntityExtension is derived from the generic class, whereby TEntity defines which entity is to be extended.

example

```
public class CompanyExtension : EntityExtension<Company>
{
    // Here is the definition of the extension properties
}
```

In the simplest case, the extended property is a simple string that is to be stored in the database under the property name. The column must be in the table to which the extended entity was mapped.

example

```
public class CompanyExtension : EntityExtension<Company>
{
    public string CreditLimit
    {
        get { return GetValue<string>("CreditLimit"); }
        set { SetValue("CreditLimit", value); }
    }
}
```

In this case, a CreditLimit column must be added to the Crm.Contact table via DB migration. The associated NHibernate mapping is generated automatically in the NHibernateInitializer.AddExtensionDynamicComponentMappings method. The method calls GetValue and SetValue within the getter and setter method are necessary to save the assigned values in a dictionary, which is required by NHibernate.

The properties can be given different attributes:

- **DatabaseAttribute** The DatabaseAttribute has two properties: ColumnName and Ignore. Using ColumnName, the property can be mapped to a DB column whose name is not identical to the property name.

example

```
[Database(Column = "CompanyKreditlimit")]
public string CreditLimit
{
    get { return GetValue<string>("CreditLimit"); }
    set { SetValue("CreditLimit", value); }
}
```

If you set Ignore = true, the property for the mapping is completely ignored. This is necessary, for example, for lookup properties (see below), which appear as a dropdown list on the UI, but should not be mapped to the DB, since only the lookup key is saved for lookups.

example

```
public class CompanyExtension : EntityExtension<Company>
{
    [LookupKey]
    public string IndoorServiceKey
    {
        get { return GetValue<string>("IndoorServiceKey"); }
        set { SetValue("IndoorServiceKey", value); }
    }

    [Database(Ignore = true)]
    public IndoorService IndoorService {
        get
        {
            return LookupManager.Instance.Get<IndoorService>(IndoorServiceKey);
        }
    }
}
```

- **LookupKeyAttribute** The LookupKeyAttribute identifies a lookup key. Properties with this attribute are not rendered on the surface, but are filled via the drop-down list of the associated lookup property. According to the naming convention, the name of the lookup key property must correspond to the name of the lookup property with the extension "Key" (see example above).
- **LookupFilterAttribute** This attribute can be used to filter the list of lookup values in the drop-down list. The attribute has two properties:

MethodName NMethodName Name of a property of the type Func <bool, TLookup>, which must be implemented within the EntityExtension class.

LookupType Type of the lookup property to be filtered.

Beispiel

```
public class CompanyExtension : EntityExtension<Company>
{
    [LookupKey]
    public string VisitFrequencyTimeUnitKey
    {
        get { return GetValue<string>("VisitFrequencyTimeUnitKey"); }
        set { SetValue("VisitFrequencyTimeUnitKey", value); }
    }

    [LookupFilter(MethodName = "OnlyTimeUnitsWithTimeUnitsPerYear",
        LookupType = typeof(TimeUnit))]
    [Database(Ignore = true)]
    public TimeUnit VisitFrequencyTimeUnit
    {
        get
        {
            return LookupManager.Instance
                .Get<TimeUnit>(VisitFrequencyTimeUnitKey);
        }
    }

    // Methods
    public bool OnlyTimeUnitsWithTimeUnitsPerYear(TimeUnit timeUnit)
    {
        return timeUnit.TimeUnitsPerYear.HasValue;
    }
}
```

```
}
}
```

- **ReadOnlyExtensionPropertyAttribute** Properties with this attribute are mapped to the DB, but cannot be edited, ie no input fields are rendered. The columns must be filled in some other way, e.g. by importing data. They are pure display fields. For examples see View CustomerCard.ascx in the JaegerLacke plugin.
- **UIAttribute** This attribute can be used to influence the display of the input field. The attribute has the following properties:

Caption Defines the resource key which is used for localizing the label of the input field, i.e. which is transferred to the Html.Localize method. Without this specification, the property name is used as the resource key.

Row Extension Properties with the same Row value are rendered on one row.

ColumnsPerRow Specifies how many columns the row is divided into. This can be used to influence the size of the input fields

Example:

```
public class CompanyExtension : EntityExtension<Company>
{
    [UI(Caption = "VisitsNeeded", Row = 1, ColumnsPerRow = 4)]
    public int VisitFrequencyValue
    {
        get { return GetValue<int>("VisitFrequencyValue"); }
        set { SetValue("VisitFrequencyValue", value); }
    }
}
```

Extended information

Visit Frequency

Years



Indoor Service

Save

[Cancel](#)

Company Extension Properties im Editor

6.9 Validation

Before an entity is stored in the database, it must be validated. There are two types of validation errors:

- Attachment failure
- Business Rules Violation

Default

Dashboard Contacts Tasks Projects Campaigns Service 56-334-34... 500004 Derby Cycl... 500001 500002

Add a Company

Basic Information

Name * 120

Name required

Company - Type Prospect Language English

Responsible User Michael Schellke Source type *

Source type required

Please choose a source to display available marketing campaigns (in execution now or closed in the last 60 days)

Address Information

Address Type * Work

Name 1

[Add Name Line](#)

Street

Zip Code * City *

Zip Code required **City required**

[Add P.O. Box](#)

Editor mit Validierungsfehlern

A binding error occurs, for example, if an attempt is made to enter a string "abc" in the text field of a property of the type int. If an entity is to be validated for binding errors, a `ModelBinder` must be implemented for this entity, which is derived from `CrmModelBinder`:

```
[ModelBinderFor(typeof(Company))]
public class CompanyModelBinder : CrmModelBinder
{
    public override object BindModel(ControllerContext controllerContext,
        ModelBindingContext bindingContext)
    {
        {
            var company = base.BindModel(controllerContext, bindingContext) as Company;
            return company;
        }
    }
}
```

A format "hh [: mm]" is expected for properties of the `TimeSpan` or `Nullable` type (information in square brackets is optional). If the permissible value is to be restricted to a range between 0 and 24 hours, the corresponding property must be provided with the `DayRestriction` attribute:

```
public class Visit : EntityBase<int>
{
    public virtual int CompanyId { get; set; }
    public virtual string CompanyName { get; set; }
    public virtual DateTime Date { get; set; }
    [DayRestriction]
    public virtual TimeSpan? Time { get; set; }
}
```

Binding errors are added to the `BindingRuleViolations` collection of the `EntityBase` class.

For the review of the business rules, classes are to be introduced which are derived from the generic class `Rule`, where `TEntity` indicates the entity to which the business rule relates. The `Rule` class defines three abstract or virtual (overwritable) methods:

```
public abstract class Rule<TEntity> : Rule
    where TEntity : class
{
    protected abstract RuleViolation CreateRuleViolation(TEntity entity);
    public abstract bool IsSatisfiedBy(TEntity entity);

    protected virtual bool IsIgnoredFor(TEntity entity)
    {
        {
            return false;
        }
    }
}
```

`IsIgnoredFor` defines whether this business rule should be ignored for the current entity. This is the case, for example, for notes that are not assigned to any entity. In this case, it is not necessary to check for the existence of the `ContactId`:

```
public class ContactIdMustExist : Rule<Note>
{
    protected override bool IsIgnoredFor(Note note)
    {
        return note.ContactId == null;
    }
    ...
}
```

IsSatisfiedBy implements the actual business logic and should return true if the current entity is valid, false otherwise:

```
public class ContactIdMustExist : Rule<Note>
{
    private readonly IContactService contactService;

    public override bool IsSatisfiedBy(Note note)
    {
        return contactService.DoesContactExist(note.ContactId.Value);
    }
}
```

The CreateRuleViolation method creates an object of the RuleViolation type if the business rule returns a validation error. An overload of the RuleViolation method can be used to create this object:

```
public class ContactIdMustExist : Rule<Note>
{
    protected override RuleViolation CreateRuleViolation(Note note)
    {
        return RuleViolation(note, n => n.ContactId, "Contact");
    }
}
```

The error message can be controlled via the ErrorMessageKey or RuleClass properties of the RuleViolation class. If there is an ErrorMessageKey, the resource file is searched for a key "RuleViolation.", Eg "RuleViolation.AccountInactive". If there is no ErrorMessageKey, the RuleClass is used to generate the key, eg "RuleViolation.Required".

The RuleClass is passed to the constructor of the Rule base class. This is a value of the RuleClass enumeration:

```
public class ContactIdMustExist : Rule<Note>
{
    public ContactIdMustExist(IContactService contactService)
        : base(RuleClass.MustExist)
    {
        this.contactService = contactService;
    }
}
```

For frequently used business rules, there are already some base classes that make implementation easier. These can be found in the namespace CrmLibrary.Validation.BaseRules of CrmLibrary.dll.

Example:

```
public class TextRequired : RequiredRule<Note>
{
    public TextRequired()
    {
        Init(n => n.Text);
    }
}
```

6.10 Client-side validation

By default, the complete validation takes place on the server side, ie the user entries are sent to the server, where the validation rules are checked and if a rule is violated, an HTML page with the error messages is rendered and sent back to the client. To save yourself this server round trip, there is the option of activating a client-side validation based on the jquery.validate.js plug-in using the HtmlHelper extension method ClientSideValidation.

Note: Client-side validation has so far only been implemented for certain, frequently occurring rules (e.g. RequiredRule, MaxLengthRule, etc.). In addition, a round trip to the server is always necessary for certain rules, for example to check the uniqueness of a name in the database.

The ClientSideValidation method has several overloads.

```
string ClientSideValidation<TEntity>(this IHttpHelper htmlHelper)

string ClientSideValidation<TEntity>(this IHttpHelper htmlHelper,
                                     string formSelector)
```

In the first variant, the HTML page is searched for a FORM tag whose fields are validated. The second overload enables a restriction to a specific form with the help of a jQuery selector:

```
<%= Html.ClientSideValidation<Person>(".person-edit") %>
```

A JavaScript tag is rendered into the page for client-side validation. There is a template Content / js / templates / validation.js for this. Valid JavaScript is generated from this using the following code within the ClientSideValidation method:

```
var sb = new StringBuilder();
sb.AppendLine("<script type=\"text/javascript\">");
sb.Append(TemplateProvider.GetTemplate("Content/js/templates/validation.js"));
sb.AppendLine("</script>");
sb.Replace("[[form_selector]]", formSelector);
sb.Replace("[[rules_as_json]]", rulesAsJson.ToString(",");
sb.Replace("[[messages_as_json]]", messagesAsJson.ToString(",");

return sb.ToString();
```

RulesAsJson and messagesAsJson are Json objects that describe the rules and associated error messages. These are required by the jquery.validate plugin.

6.11 Overriding business rules

A plugin can override a business rule for an entity. The override is based on the entity type, class name and priority. The default priority is 0.

Example:

The main program defines a business rule for the Name property of the project entity:

```
public class NameMaxLength : MaxLengthRule<Project>
{
    public NameMaxLength()
    {
        Init(c => c.Name, 120);
    }
}
```

A plugin can override this rule with the following class:

```
public class NameMaxLength : MaxLengthRule<Project>
{
    public override int Priority
    {
        get { return 10; }
    }

    public NameMaxLength()
    {
        Init(c => c.Name, 20);
    }
}
```

6.12 Background agents

Background agents are background processes that run at regular time intervals. They must derive from the base class BackgroundServiceBase and implement the abstract method Run and be provided with the attribute [BackgroundService]. The Run method contains the logic of the background agent and is called periodically. The standard time interval is 5 minutes. If a different time interval is to be specified, this must be defined in the OnInitializeSettings method. Additional configuration data can also be specified here, which are required by the background agent. These specifications are written into the DB table BackgroundServiceSetting during the first execution and can be changed later on the admin page of the Crm application.

Example

```
[BackgroundService]
public class MessageSender : BackgroundServiceBase
```

```

{
    protected override void Run()
    {
        SendMessages();
    }

    protected override void OnInitializeSettings()
    {
        SetSetting("Interval", TimeSpan.FromSeconds(61).TotalSeconds.ToString());
        SetSetting("Host", "127.0.0.1");
        SetSetting("Port", "25");
    }

    ...
}

```

Note: If the Run method is to be debugged when the application is started, the `RunOnFirstStartUp` property must be set to true in the constructor. In this case, the Run method is executed synchronously when the program is started (ie not as a background thread).

6.13 FAQ

How is a plugin activated / deactivated? Via the site settings or site settings, plugins can be activated and deactivated under the Properties or Properties tab.

Where are the activated plugins stored in the database? In the table `Crm.Site`. The `ActivePlugins` column contains the names of the active plugins in the form of a comma-separated list.

How are the DB structures required by a plugin created? When a plugin is activated for the first time, all create scripts in the Database / Create subdirectory of the plugin are executed. The scripts are executed in the lexical order of their filename. A `SchemalInfo` table for the plugin is also created in which the migrations are saved. After running the create scripts, the migrations are carried out in the Database / Migrate folder and their migration number is saved in the `SchemalInfo` table. Each time the plugin is activated again, only those migrations are carried out whose migration number is not in the `SchemalInfo` table. In other words, if you have written a new migration and want to use it, the plug-in must be deactivated and then reactivated in order to execute the migration.

Note The check whether the plug-in is being activated for the first time is carried out using the plug-in's `SchemalInfo` table. The create scripts are only executed if this table does not exist.

How are Selenium tests for plugins written? In order to be able to test a plugin, the plugin must be activated. The `IsPluginActive`, `ActivatePlugin` and `DeactivatePlugin` methods are available in the `SeleniumTest` class for this purpose. The setup method should check whether the plug-in to be tested is active. If not, it should be activated by calling `ActivatePlugin`.

Example:

```

[TestFixture]
public class ExpenseTests : SeleniumTest
{
    [SetUp]
    public void Setup()
    {
        if (!IsPluginActive("Crm.Service"))
        {
            ActivatePlugin("Crm.Service");
        }
        GoToUrl("/Crm.Service/Expense", DefaultUserToken);
        ...
    }
}

```

How is a new note type defined in a plugin? Note types must derive from the base class `Note` and override the `IsSystemGenerated` property. For notes generated by the system (e.g. when completing a task) this property should return the value true, otherwise false. System-generated notes are marked by a rectangle with rounded corners and an identifier text (eg "Task" or "Task"). The color of the rectangle and the text are set using the `ImageColor` and `ImageTextKey` properties. For non-system-generated notes, a url for an image must be specified using the `ImageVortualUrl` property

Example

```

public class OrderStatusChangedNote : Note
{
    public override bool IsSystemGenerated { get { return true; } }
}

```

```

    public override string ImageTextKey { get { return "Status"; } }
}

```

A subclass mapping for the new note type must be specified as the NHibernate mapping. The different note types are differentiated using the DiscriminatorValue.

```

public class OrderStatusChangedNoteMap : SubclassMapping<OrderStatusChangedNote>
{
    public OrderStatusChangedNoteMap()
    {
        DiscriminatorValue("OrderStatusChanged");
    }
}

```

How can a lookup be extended by a plugin? If a lookup is to be expandable by plugins and the lookup values are to be accessed in the code (i.e. the lookup values are known at development time and are not created by the user via the user interface), the value of a lookup object can be checked using static methods.

Example

```

[Lookup("[SMS].[ServiceOrderDispatchStatus]")]
public class ServiceOrderDispatchStatus : Lookup<string>
{}

// Extension methods
public static class ServiceOrderDispatchStatusExtensions
{
    public static bool IsScheduled(this ServiceOrderDispatchStatus status)
    {
        return status.Key == "Scheduled";
    }

    public static bool IsReleased(this ServiceOrderDispatchStatus status)
    {
        return status.Key == "Released";
    }
}

```

The lookup values "Scheduled" and "Released" are known at development time. If a plugin introduces a new value, a new extension method can be implemented in the plugin:

```

public static class PluginServiceOrderDispatchStatusExtensions
{
    public static bool IsInSomeNewStatus(this ServiceOrderDispatchStatus status)
    {
        return status.Key == "IsInSomeNewStatus";
    }
}

```

The advantage of this procedure compared to the definition of static properties in the main program is that lookup values defined in the plugin can be addressed with the same syntax as the lookup values defined in the main program. If the lookup values of the main program are defined with the help of static properties, this would not be possible because the lookup class cannot be extended.

7 Multitenancy

With 5.0 support for multiple tenants was added to the product. There are two modes which are supported - physical tenants and logical tenants. See this blog post to understand the differences: [Thoughts on 5.0 and our upcoming API release](#). This chapter focuses on logical tenants (InstancePerTenant=false).

7.1 Database

Currently we use only a small subset of Unicore to manage tenant / domain access of users. You can manage this access on entity level, operations like reading and writing and entity types. To keep things relatively easy we just use one operation (RW) and handle access only on entity type level.

7.1.1 dbo.Domain

This will be the tenant itself (formerly CRM.Site). There is always a default domain using empty guid (00000000-0000-0000-0000-000000000000) as key.

7.1.2 dbo.DomainAuthorisedDomain

Can be used for a hierarchical domain structure. Currently every domain needs an entry referencing itself here.

7.1.3 dbo.User

This table is needed for Unicore filtering which user belongs to which domain and which entity types can the user access in which domain. The table references CRM.User via a foreign key. Currently we did not merge CRM.User and dbo.User to one table, but this will be done in an upcoming release.

7.1.4 dbo.EntityType

Here every type that needs to be authorized (meaning having it's access checked via special filters) needs an entry.

7.1.5 dbo.EntityAccess

This table contains the access definition which will then be granted in dbo.GrantedEntityAccess. In our use case dbo.EntityAccess contains one entry for every entity type.

7.1.6 dbo.EntityAuthData

Here our authorized entities are connected to Unicore and their owning domain is saved. Each authorized entity has a foreign key AuthDataId which points to dbo.EntityAuthData.

7.1.7 dbo.GrantedEntityAccess

This table connects user, domain and access. In our case this means it determines which user can access which entity type in which domain.

7.2 Authorization Filter

If the tenant plugin is active and running in logical tenant mode, this filter is enabled in all sessions for all authorized entites and also collections of authorized entities. It is rather complex and uses the tables described above to decide which data the current user can access. There are five parameters which will be set for each session, This is done in UnicoreAuthorisationFilterEnabler: - User: the current user's id - Operation: always set to RW - DomainType: always set to Normal - NoDomainFilter: always set to false - Domains: set to the common domain **and** the current domain

7.3 Services and Interfaces

7.3.1 NHibernate Mappings

The NHibernate mappings are applied dynamically, except for the filter itself. The mapping is automatically applied to types inheriting `EntityBase<Guid>`. There are also two marker interfaces to explicitly include or exclude a type: `IExplicitlyAuthorizedObject` and `INoAuthorisedObject`.

7.3.2 SetEntityAuthDataEventHandler

This event handler will automatically add the required Unicore database entries for the filtering to work when new entities are inserted into the database.

7.3.2.1 IDomainForTypeProvider

If there is an implementation of this interface where `T` is an authorized entity, it will be called to determine the domain of the newly created entities of type `T` on `PreInsert`, else the default implementation will be called. The returned domain does not have to be the current domain, as you might want some types to be always inserted into the common domain. For example, check `DomainForUserProvider`. Also, the implementation can cover multiple entities in a single class like this sample:

```
public class DomainForUserAndUsergroupProvider : IDomainForTypeProvider<Crm.Library.Model.User>, IDomainForTypeProvider<Crm.Library.Model.Usergroup>
{
    public Guid GetDomain(Crm.Library.Model.User entity) => UnicoreDefaults.CommonDomainId;
    public Guid GetDomain(Crm.Library.Model.Usergroup entity) => UnicoreDefaults.CommonDomainId;
    //In case of multiple interface implementations, this is needed or we get a compiler error.
    public Guid GetDomain(IAuthorisedObject entity) =>
    {
        entity switch
        {
            Crm.Library.Model.User => GetDomain((Crm.Library.Model.User)entity),
            Crm.Library.Model.Usergroup => GetDomain((Crm.Library.Model.Usergroup)entity),
            _ => throw new NotImplementedException()
        };
    }
}
```

7.3.2.2 IIdForEntityProvider

This provider returns the id of an entity, this is especially important for types which were explicitly marked with `IExplicitlyAuthorizedObject` and do not return an id of type `Guid` by default (as an `EntityBase<Guid>` would).

7.4 Migrations

Migrations are running in this order (this is just an overview): - add table `dbo.Domain`, a common domain with `Id` empty `Guid` will be added as default and `CRM.Site` migrated - add table `dbo.EntityType` - add tenant types (`*_AddEntityType.cs`) - these migrations are added to every plugin that defines authorized entities - migrate `CRM.Tenant`, which will become entries in `dbo.Domain` - add table `dbo.User`, migrate `CRM.UserTenant` - add other required Unicore tables - migrate column `TenantKey`, add entries to `dbo.EntityAuthData`, depending on the entries in `dbo.EntityType` - add a new column `AuthDataId` to the target table referencing `dbo.EntityAuthData` - add the entry in `dbo.EntityAuthData` - add entries in `dbo.GrantedEntityAccess` for the entity type for every user and every domain the user has access to

If you have custom entities in your customer plugin that need to be authorized, you must add a migration. The easiest way is to use `AddOrUpdateEntityAuthDataColumn<T>([...])` of `UnicoreMigrationHelper`.

7.5 Imports

7.5.1 User

For your user import you just have to make sure to add an entry to `CRM.User` and `dbo.User`.

7.5.2 Domain / Tenants

Import to `dbo.Domain`, don't forget to add an entry to `dbo.DomainAuthorizedDomain`

7.5.3 Entities

Make sure your types are registered, then you only have to add an entry to `dbo.EntityAuthData`. To do so you should output the merge action, then make an insert to `dbo.EntityAuthData` for all entities that were inserted and finally update `AuthDataId` in the merge target.

Working example importing one note into the default domain:

```
IF OBJECT_ID('tempdb..#Source') IS NOT NULL DROP TABLE #Source
IF OBJECT_ID('tempdb..#Imported') IS NOT NULL DROP TABLE #Imported
IF OBJECT_ID('tempdb..#AuthData') IS NOT NULL DROP TABLE #AuthData
CREATE TABLE #Imported ([Action] NVARCHAR(100), EntityId UNIQUEIDENTIFIER, LegacyId NVARCHAR(100))
CREATE TABLE #AuthData (AuthDataId UNIQUEIDENTIFIER, EntityId UNIQUEIDENTIFIER)

SELECT NEWID() AS [LegacyId], 'Text' AS [Text], CONVERT(UNIQUEIDENTIFIER, '00000000-0000-0000-0000-000000000000')
INTO #Source

DECLARE @entityTypeId UNIQUEIDENTIFIER = (SELECT [Uid] FROM [dbo].[EntityType] WHERE [Name] = 'Crm.Model.UserNote')

BEGIN TRANSACTION
    MERGE [CRM].[Note] AS [target]
    USING #Source AS [source]
    ON [target].[LegacyId] = [source].[LegacyId]
    --WHEN MATCHED [...]
    WHEN NOT MATCHED THEN INSERT ([Text], [LegacyId]) VALUES ([source].[Text], [source].[LegacyId])
    --WHEN NOT MATCHED BY SOURCE [...]
    OUTPUT $action AS [Action], INSERTED.[NoteId] AS [EntityId], INSERTED.[LegacyId] INTO #Imported;

    INSERT INTO [dbo].[EntityAuthData] ([EntityId], [EntityTypeId], [DomainId])
    OUTPUT INSERTED.[Uid] AS AuthDataId, INSERTED.EntityId INTO #AuthData
    SELECT [imported].[EntityId], @entityTypeId, [source].[DomainId]
    FROM #Imported [imported]
    JOIN #Source [source] ON [source].[LegacyId] = [imported].[LegacyId]
    WHERE [imported].[Action] = 'INSERT'

    UPDATE [target]
    SET [target].[AuthDataId] = [authData].[AuthDataId]
    FROM #AuthData [authData]
    JOIN [CRM].[Note] [target] ON [authData].EntityId = [target].NoteId
COMMIT TRANSACTION
```

8 Our JS globalization story

8.1 Why do we need globalization?

Each language, and the countries that speak that language, have different expectations when it comes to how numbers (including currency and percentages) and dates should appear. Obviously, each language has different names for the days of the week and the months of the year. But they also have different expectations for the structure of dates, such as what order the day, month and year are in. In number formatting, not only does the character used to delineate number groupings and the decimal portion differ, but the placement of those characters differ as well.

A user using an application should be able to read and write dates and numbers in the format they are accustomed to. This library makes this possible, providing an API to convert user-entered number and date strings - in their own format - into actual numbers and dates, and conversely, to format numbers and dates into that string format.

Even if the application deals only with the English locale, it may still need globalization to format programming language bytes into human-understandable language and vice-versa in an effective and reasonable way. For example, to display something better than "Edited 1 minutes ago".

8.2 Reference

[jquery/globalize on github](#)

8.3 General

The culture is initialized by the browser's default culture at startup of the application in *Helper.Culture.js*. For further operations as parsing this culture is used by default. If an other cultures deviating from the loaded culture should be required the specific CLDR data have to be loaded.

[Cldr.js on github](#)

8.4 Formatting dates and time

8.4.1 Parameter

value Date instance to be formatted, eg. `new Date()`;

8.4.1.1 Options

A JSON object including one of the following.

skeleton String value indicating a skeleton (see description above), eg. `{ skeleton: "GyMMMd" }`. Skeleton provides a more flexible formatting mechanism than the predefined list full, long, medium, or short represented by date, time, or datetime. Instead, they are an open-ended list of patterns containing only date field information, and in a canonical order.

date One of the following String values: full, long, medium, or short, eg. `{ date: "full" }`.

time One of the following String values: full, long, medium, or short, eg. `{ time: "full" }`.

datetime One of the following String values: full, long, medium, or short, eg. `{ datetime: "full" }`

raw String value indicating a machine raw pattern eg. `{ pattern: "dd/mm" }`. *Note:* This is NOT recommended for i18n in general. No globalization can be done by this pattern. Use skeleton instead.

8.4.2 Examples

The following examples demonstrates a comparison of the former globalization version (v0.0.1) to the new one (v1.0.0).

8.4.3 "01.09."

- *Old:* no globalization pattern
- *New:* `Globalize.formatDate(new Date(), { skeleton: 'MMdd' })`

8.4.4 "01.09.15"

- *Old*: no globalization pattern
- *New*: `Globalize.formatDate(new Date(), { date: "short" })`

8.4.5 "01.09.2015"

- *Old*: `Globalize.format(new Date(), "d")`
- *New*: `Globalize.formatDate(new Date(), { date: "medium" })`

8.4.6 "Dienstag, 1. September 2015"

- *Old*: `Globalize.format(new Date(), "D")`
- *New*: `Globalize.formatDate(new Date(), { date: 'full' })`

8.4.7 "09:08"

- *Old*: `Globalize.format(new Date(), "t")`
- *New*: `Globalize.formatDate(new Date(), { time: 'short' })`

8.4.8 "Dienstag, 1. September 2015 09:08"

- *Old*: `Globalize.format(new Date(), "f")`
- *New*: `Globalize.formatDate(new Date(), { skeleton: 'yMMMEdHm' })`

Note: New format is slightly different: "Di., 1. Sep. 2015, 09:08"

8.5 Formatting numbers

8.5.1 Parameter

value A string number.

8.5.1.1 Options (optional)

A JSON object including none or any of the following options.

style Optional String decimal (default), or percent.

minimumIntegerDigits Optional Non-negative integer Number value indicating the minimum integer digits to be used. Numbers will be padded with leading zeroes if necessary.

minimumFractionDigits and maximumFractionDigits Optional Non-negative integer Number values indicating the minimum and maximum fraction digits to be used. Numbers will be rounded or padded with trailing zeroes if necessary. Either one or both of these properties must be present. If they are, they will override minimum and maximum fraction digits derived from the CLDR patterns.

minimumSignificantDigits and maximumSignificantDigits Optional Positive integer Number values indicating the minimum and maximum fraction digits to be shown. Either none or both of these properties are present. If they are, they override minimum and maximum integer and fraction digits. The formatter uses however many integer and fraction digits are required to display the specified number of significant digits.

round Optional String with rounding method ceil, floor, round (default), or truncate.

useGrouping Optional Boolean (default is true) value indicating whether a grouping separator should be used.

8.5.2 Plain Numbers

```
Globalize.formatNumber(1234.567) // > "1.234,567"
```

```
Globalize.formatNumber(1234.567, {minimumSignificantDigits: 3, maximumSignificantDigits: 5}) // > "1.234,6"
```

```
Globalize.formatNumber(12, {minimumSignificantDigits: 3, maximumSignificantDigits: 5}) // > "12,0"
```

```
Globalize.formatNumber(1234.567, {minimumFractionDigits: 1, maximumFractionDigits: 2}) // > "1.234,57"
```

```
Globalize.formatNumber(1234, {minimumFractionDigits: 1, maximumFractionDigits: 2}) // > "1.234,0"
```

8.5.3 Percentage

```
Globalize.formatNumber(0.0123456789, {style: "percent", minimumSignificantDigits: 3, maximumSignificantDigits: 5}) // > "1,2346 %"
```

8.6 Parsing dates

Return a function that parses a string representing a date into a JavaScript Date object according to the given options. The default parsing assumes numeric year, month, and day (i.e., { skeleton: "yMd" }). The returned function is invoked with one argument: the String value to be parsed.

8.6.1 Parameters

value String with date to be parsed, eg. "11/1/10, 5:55 PM"

8.6.1.1 Options (optional)

See `.dateFormatter([options])`.

8.6.2 Examples

As described above, by default the initialized culture is used for parsing. If another culture is required it has to be loaded by Cldr.

```
Globalize.parseDate("01.02.2015") // > Sun Feb 01 2015 00:00:00 GMT+0100 (Mitteleuropäische Zeit)
```

8.7 Parsing numbers

Return a function that parses a String representing a number according to the given options. If value is invalid, NaN is returned. The returned function is invoked with one argument: the String representing a number value to be parsed.

8.7.1 Parameters

value String with number to be parsed, eg. "3.14".

8.7.1.1 Options (optional)

A JSON object including none or any of the following options.

style Optional String decimal (default), or percent.

8.7.2 Examples

```
Globalize.parseNumber("12.735,00") // > 12735
```

```
Globalize.parseNumber("6.626E-34") // > 6.626e-31
```

8.8 Calendar constants

```
Globalize.cldr.main("dates").fields.day["relative-type-0"] // > "heute"
```

```
Globalize.cldr.main("dates").fields.month.displayName // > "Monat"
```

```
Globalize.cldr.main("dates").fields.week.displayName // > "Woche"
```

```
Globalize.cldr.main("dates").fields.day.displayName // > "Tag"
```

9 Moment.js for date/time operations

9.1 Preamble

Moment.js is exclusively used for date/time operations. Formatting date/time strings culture depending is not scope of moment.js.

You can find a short introduction with a few examples in this document.

9.2 Reference

You can find detailed development docs on <http://momentjs.com>.

9.3 Examples

9.3.1 Now

```
var now = moment();
```

This is essentially the same as calling `moment(new Date())`.

9.3.2 Add

```
moment().add(7, 'days');
```

There are some shorthand keys.

```
moment().add(7, 'd'); // d = days
```

Note: Please do **not** use shorthand keys for better readability.

9.3.3 Start of Time

```
moment().startOf(String);
```

Mutates the original moment by setting it to the start of a unit of time.

```
moment().startOf('year'); // set to January 1st, 12:00 am this year
moment().startOf('month'); // set to the first of this month, 12:00 am
moment().startOf('week'); // set to the first day of this week, 12:00 am
moment().startOf('day'); // set to 12:00 am today
```

9.3.4 Format

```
moment().format(String);
```

Format can be used for static date/time formats.

```
moment().format("YYYY-MM-DD");
```

Note: Do not use moment.js for globalization!

10 Custom knockout binding handlers

You're not limited to using knockouts built-in bindings like `click`, `value`, and so on, there are already some custom binding handlers waiting for you.

10.1 autosize

The `autosize` binding causes the associated `<textarea>` element to automatically adjust its height on keyboard or window resize events.

10.1.1 Example

```
html
<textarea class="form-control fg-input" data-bind="value: value, autosize: true"></textarea>
```

10.1.2 Parameters

- Main parameter

A value that controls whether or not the associated `textarea` element automatically adjust its height.

Non-boolean values are interpreted loosely as boolean. For example, `0` and `null` are treated as `false`, whereas `21` and non-`null` objects are treated as `true`.

If your parameter references an observable value, the binding will update the `autosize` behavior whenever the observable value changes. If the parameter doesn't reference an observable value, it will only set the state once and will not do so again later.

10.2 colorPicker

The `colorPicker` binding is used to enhance an element with the [Bootstrap Colorpicker](#).

10.2.1 Example

```
html
<button class="btn btn-default btn-sm waves-effect" data-bind="colorpicker: myObservable">
  <i class="zmdi zmdi-palette"></i>
</button>
```

10.2.2 Example with additional colorpicker options

```
html
<button class="btn btn-default btn-sm waves-effect" data-bind="colorpicker: { value: myObservable, options: { align: 'left' } }">
  <i class="zmdi zmdi-palette"></i>
</button>
```

10.2.3 Parameters

- Main parameter

The main parameter should be an observable with a value of either `null`, or a color as hexadecimal string. Whenever a color from the colorpicker is selected, the observable will be updated with the hexadecimal string representing the selected color. Optionally the parameter can be an object with a `value` and an `options` property to set any of the [available configuration options](#) to the colorpicker.

10.3 datePicker

The `datePicker` binding is used to enhance an `<input>` element with the [Bootstrap Datepicker](#). The binding handler will utilize [Moment.js](#) to format and parse Date objects.

10.3.1 Example


```
html
<input type="text" class="form-control date-picker" data-bind="datePicker: dateFrom" placeholder="Click here">
<input type="text" class="form-control date-picker" data-bind="datePicker: dateFrom, datePickerOptions: { pickTime: true }" />
<input type="text" class="form-control date-picker" data-bind="datePicker: dateFrom, datePickerOptions: { pickTime: true, onlyTime: true }" />
<input type="text" class="form-control date-picker" data-bind="datePicker: dateFrom, datePickerOptions: { pickDateTime: true }" />
<input type="text" class="form-control date-picker" data-bind="datePicker: dateFrom, datePickerOptions: { keepTimeZone: keepTimezone }" />
<input type="text" class="form-control date-picker" data-bind="datePicker: dateFrom, datePickerOptions: { config: { minDate: minDate, maxDate: maxDate } }" />
<form-element params="value: From, caption: 'From', type: 'timePicker', datePickerOptions: { pickDateTime: true, config: { minDate: minDate, maxDate: maxDate } }" />
```

10.3.2 Parameters

- Main parameter

The main parameter should be an observable with a value of either null, a Date object or a timespan string. Whenever a date from the datepicker is selected, the observable will be updated with the Date object representing the selected date, null if the selection was cleared, or the timespan as string when a duration is selected.

- datePickerOptions

The parameter datePickerOptions is a plain object with the following properties: - pickTime set to true provides an easy way to pick a time (without a date) - pickDateTime set to true provides an easy way to pick a datetime (a date and time combined) - pickDuration set to true provides an easy way to pick a duration (like pickTime but bound to an observable which contains the duration as a timespan string) - onlyTime set to true provides an easy way to pick a time (without a date). Difference between pickTime and onlyTime is if this property is set to true, then stored value will contain only time in hh:mm format - config allows you to set parameters to datepicker directly, please refer to [Bootstrap Datepicker Options](#) - keepTimeZone set to true prevents timezone info from being stripped when picking dates without time

10.4 dateRange

The dateRange binding is used to display a formatted range of two Date objects as any element's text (e.g. h1, p, a, span, etc.), omitting redundant month and year information. The binding handler utilizes [Globalize.js](#) to format the Date objects.

10.4.1 Example

```
html
<small data-bind="dateRange: {
start: moment(selectedDate()).startOf('isoWeek'),
end: moment(selectedDate()).endOf('isoWeek')
}"></small>
```

10.4.2 Parameters

- Main parameter

The main parameter should be an object with the following properties:

- start
 - A date or moment object used as the start of the date range.
- end
 - A date or moment object used as the end of the date range.

10.5 dateText

The dateText binding is used to display a formatted Date object as any element's text (e.g. h1, p, a, span, etc.). The binding handler utilizes [Globalize.js](#) to format the Date objects.

10.5.1 Example

```
html
<span data-bind="dateText: OrderDate"></span>
```

10.5.2 Example 2: Specifying a custom pattern

```
html
<span data-bind="dateText: { value: new Date(), pattern: { time: 'short' } }">
```

10.5.3 Example 3: Use time ago format

```
html
<span data-bind="dateText: { value: CreateDate, pattern: { datetime: 'medium' }, useTimeAgo: 'Day' }"></span>
```

10.5.4 Parameters

- Main parameter

The binding handler sets the text of the element to a formatted string of the parameters date. If the parameter is an observable value, the binding will update the element's text whenever the value changes. If the parameter isn't observable, it will only set the text once and will not update it again later.

The default pattern the date will be formatted is date: 'medium' which is for example dd.MM.yyyy or MM/dd/yyyy. To specify a custom pattern, a object can be used as the main parameter with the following properties:

- value

The Date or an observable with a Date value, see above

- pattern

A pattern supported by the [Globalize.js date formatter](#)

- defaultText

A String that will be rendered if value is null or undefined.

- timeZone

This timezone is used to convert the provided value to a timezone. Format is IANA timezone name, eg. *Europe/Berlin*.

- useLocal

In case this boolean value is specified true, the timezone will be ignored.

- useTimeAgo

When using this parameter it will give back a time ago string within the period given as value to the parameter. The date values out of this boundary will have default date format.

Values that can be used for this parameter:

- 'Hour': Gives back dates in time ago format within 1 hour to the current date
- 'Day': Gives back dates in time ago format within 1 day to the current date
- 'Week': Gives back dates in time ago format within 1 week to the current date
- 'Month': Gives back dates in time ago format within 1 month to the current date
- 'Year': Gives back dates in time ago format within 1 year to the current date

Any other value given to this parameter will result in a full time ago string (no limit, uses x years ago format at max)

10.6 dropzone

The dropzone binding is used to display a dropzone for file uploads. The user has the possibility to upload a file per drag and drop or just by clicking on the dropzone.

10.6.1 Example 1: Use the dropzone to upload one file

```
html
<div class="dropzone" data-bind="dropzone: { url: '#', fileResource: FileResource, acceptFile: 'application/pdf', maxFiles: 1 }">
```

10.6.2 Example 2: Use the dropzone to upload multiple files

```
html
<div class="dropzone" data-bind="dropzone: { url: '#', files: FileResources, acceptFile: 'application/pdf', maxFiles:null }">
```

10.6.3 Parameters

For the use of the drop zone there are several parameters that define the behavior of the drop zone:

- url

always should be '#' because the upload is handled in the JavaScript and there is no request to the server

- fileResource

should be defined if the user should be able to upload exactly one file. This parameter should be a file resource object.

- files

should be defined if the user should be able to upload multiple files. This parameter expects an empty array which will be filled with file resource objects.

- acceptFile

A String which defines, which files are accepted in the dropzone, e.g. application/pdf, image/jpeg, image/png, image/gif. To accept all files just pass an empty string

- maxFiles

An integer defining the maximum number of uploadable files. If this parameter is not defined, there is no limit

- name

A string which helps to identify the dropzone element. Useful when multiple dropzone elements defined in a form.

- disable

Makes the dropzone disabled, expects a boolean.

10.7 durationText

The durationText binding is used to display a formatted [moment duration object](#) or a [ISO8601 string](#) as any element's text (e.g. h1, p, a, span, etc.).

The default template the duration will be formatted is hh:mm. To specify a custom template, a object can be used as the main parameter with the following properties:

- value

The duration object or string, see above

- template

A template supported by the [Moment Duration Formatter](#)

- defaultText

A String that will be rendered if value is null or undefined.

10.7.1 Example

```
html
<span data-bind="durationText: TaskDuration"></span>
```

10.7.2 Example 2: Specifying a custom template

```
html
<span data-bind="durationText: { value: TaskDuration, template: 'hh:mm:ss' }"></span>
```

10.7.3 Parameters

- Main parameter

The binding handler sets the text of the element to a formatted string of the main parameter. If the parameter is an observable value, the binding will update the element's text whenever the value changes. If the parameter isn't observable, it will only set the text once and will not update it again later.

10.8 fileInput

The fileInput binding is a two-way binding for <input type="file" /> elements.

10.8.1 Example

```
html
<input type="file" data-bind="fileInput: {
    content: user().Avatar,
    blob: avatarBlob,
    content: user().Base64Avatar,
    name: avatarFilename,
    contentType: avatarContentType,
    size: avatarFileSize,
    compressionQualityLevel: window.Helper.Image.CompressionQualityLevel.Good
}">
```

10.8.2 Parameters

- Main parameter

The main parameter should be an object which can have any of the following optional properties:

- blob

When specified, the selected file content will be set as a blob to the parameters observable value.

- content

When specified, the selected file content will be set as base64 to the parameters observable value.

- name

When specified, the selected file name will be set to the parameters observable value.

- contentType

When specified, the selected file content type will be set to the parameters observable value.

- size

When specified, the selected file size in bytes will be set to the parameters observable value.

- compressionQualityLevel

When specified, the selected file will be compressed with this quality, otherwise 'Good' (60%) compression is the default.

10.9 fileResource

The purpose of the fileResource binding is to bind a FileResource object to a link, to either open in on the device, or be able to download / save it on the device (via HTML5s download attribute).

This binding handler changes its behavior depending on the environment: In the Cordova app for Android it will use native Android functions via cordova plugins, in the Cordova app for iOS it will use native iOS functions. If the Crm.Offline plugin is available it will check if the files are offline available and replace the href values with blob-URLs.

10.9.1 Example 1: Opening a file resource in a new tab

```
html
<a href="#" data-bind="fileResource: $data.FileResource" target="_blank">
    Open File
</a>
```

10.9.2 Example 2: Downloading a file resource

```
html
<a data-bind="fileResource: $data.FileResource, attr: { download: FileName }">
    Save File
</a>
```

10.9.3 Parameters

- Main parameter

The main parameter should be a file resource object.

If this parameter is an observable value, the binding will update whenever the observable changes.

10.10 fileSize

The purpose of the `fileSize` binding handler is to display numbers representing a file size in bytes. The formatted values will be set as the element's text. For formatting with the correct decimal separator, [Globalize.js](#) is utilized, also the formatted value will always have 2 fraction digits.

10.10.1 Example

```
html
<li>
  <span data-bind="fileSize: Length"></span>
</li>
```

10.10.2 Parameters

- Main parameter

The binding handler sets the text of the element to a formatted string of the parameters value. If the parameter is an observable, the binding will update the element's text whenever the value changes. If the parameter isn't observable, it will only set the text once and will not update it again later.

10.10.3 Note

NaN and null values will be displayed as 0,00 KB.

10.11 flotChart

The `flotChart` binding is a one-way binding for displaying objects in a chart using the [Flot](http://www.flotcharts.org/ library.

10.11.1 Example

```
```html <div data-bind="flotChart: { data: data, options: options }" class="flot-chart m-l-20 p-r-30" style="width: 100%; height: 300px;">
...
```
```

10.11.2 Parameters

- Main parameter

The main parameter should be an object with the following properties:

- data

An array of data series which must be in the [Flot data format](#).

- options

An optional options object which will be passed to the flot library, for a list of all supported options please consult the [Flot API documentation](#).

10.12 fullCalendar

The `fullCalendar` binding is a one-way binding for feeding the [FullCalendar](#) with an observable array of arbitrary objects, passing a mapping function to convert these objects to FullCalendar compatible objects.

10.12.1 Example

```
```html
...

```

### 10.12.2 Parameters

- Main parameter

The main parameter should be an object with the following properties:

- items

An array of items which should be displayed in the calendar.

If this parameter is an observableArray, the calendar will update the events whenever the observableArray changes.

- getEvent

A function mapping the passed object to a [FullCalendar Event Object](#).

- editable

Optional parameter which determines whether the events on the calendar can be modified. By default false

- firstHour

Optional parameter which determines the first hour that will be visible in the scroll pane. By default 7

- selectable

Optional parameter which allows a user to highlight multiple days or timeslots by clicking and dragging. By default false

- selectHelper

Optional parameter whether to draw a "placeholder" event while the user is dragging. By default false

- start

This parameter should be used to pass an observable, which will be updated with the first visible day in the calendar whenever the user changes the displayed view. It can be used to trigger lazy loading of displayed elements. Note that in the monthly view this is often before the 1st day of the month, because most months do not begin on the first day-of-week.

- end

This parameter should be used to pass an observable, which will be updated with the last visible day in the calendar whenever the user changes the displayed view. It can be used to trigger lazy loading of displayed elements.

## 10.13 lookupValue

The purpose of the lookupValue binding is to display the translated text of a lookup for an observable which contains the lookup key only.

### 10.13.1 Example

```
html

```

### 10.13.2 Parameters

- Main parameter

The main parameter should be a lookup key. If the parameter is an observable value, the binding will update the element's text whenever the value changes. If the parameter isn't observable, it will only set the text once and will not update it again later.

- lookups

An array containing the lookups which will be used by the binding handler to find the correct lookup text for the key.

- prefix

An optional parameter which can be used to specify a string the displayed text will be prefixed with. If the parameter is an observable value, the binding will update the displayed prefix whenever the value changes.

- suffix

An optional parameter which can be used to specify a string the displayed text will be suffixed with. If the parameter is an observable value, the binding will update the displayed suffix whenever the value changes.

- `defaultValue`

An optional parameter which can be used to specify a string that will be used in case no corresponding lookup value is found. If the parameter is an observable value, the binding will update the displayed value whenever it changes.

- `defaultValue`

An optional parameter which can be used to specify a string that will be used in case no corresponding lookup value is found. If the parameter is an observable value, the binding will update the displayed value whenever it changes.

## 10.14 map

The map binding is a one-way binding for displaying objects on a map using the [Leaflet](#) library.

### 10.14.1 Example

```
html
<div data-bind="map: {
 elements: items,
 tileLayerUrl: 'http://{s}.tile.osm.org/{z}/{x}/{y}.png'
}" class="map whitebox" style="height: 500px">
</div>
```

### 10.14.2 Parameters

- Main parameter

The main parameter should be an object with the following properties:

- `elements`

An array of elements to be displayed on the map. If the parameter is an observableArray, the map will be updated whenever the array changes. If the parameter isn't observable the binding will initialize the map with the elements once and will not update it later.

To be displayed correctly on the map, each of the elements should have at least a *Latitude* and a *Longitude* property or the latitude and longitude parameters need to be passed. A *IconName* property is optional and will fallback to *marker\_pin3*

- `tileLayerUrl`

A tile layer is required for Leaflet, otherwise it will just display a gray map. The URL from the example can be used for free.

- `latitude`

An optional parameter which can contain a function to calculate the latitude to use for an item.

- `longitude`

An optional parameter which can contain a function to calculate the longitude to use for an item.

## 10.15 mapLink

The mapLink binding is a one-way binding for displaying links to google map for an *Address* object.

### 10.15.1 Example

```
html
<a data-bind="mapLink: company().StandardAddress">Open in Google Maps
```

### 10.15.2 Parameters

- Main parameter

The main parameter should be an *Address* object or an object with at least a *Street*, *ZipCode* and *City* property.

## 10.16 money

The purpose of the money binding handler is to display numbers or strings representing a money value. The formatted values will be set as the element's text. For formatting with the correct decimal separator, [Globalize.js](#) is utilized, also the formatted value will always have 2 fraction digits.

### 10.16.1 Example

```
html

 €

```

### 10.16.2 Parameters

- Main parameter

The binding handler sets the text of the element to a formatted string of the parameters value. If the parameter is an observable the binding will update the element's text whenever the value changes. If the parameter isn't observable, it will only set the text once and will not update it again later.

### 10.16.3 Note

NaN and null values will be displayed as 0,00.

## 10.17 mProgress

The mProgress binding handler attaches a material design loading indicator using [Mprogressbar.js](#) to an element.

### 10.17.1 Example

```
html
<div data-bind="mProgress: loading">
 Hello world

```

### 10.17.2 Parameters

- Main parameter

The main parameter should be an observable with a value of either true or false. The progress bar will be displayed while the value is true and is hidden while it is false.

## 10.18 noRequiredPermission

The noRequiredPermission binding is exactly the same as the requiredPermission binding, except that it inverts the result of the permission check. For more details, see documentation for the requiredPermission binding.

### 10.18.1 Example 1

```
html
<div class="pmb-block pmo-block" data-bind="noRequiredPermission: { name: 'View', group: 'Tag' }">
 You do not have the permission to see the tags.
</div>
```

### 10.18.2 Parameters

- Main parameter:

The main parameter should be an object with the following properties:



- name

The permission name required to hide the contained markup.

- group

The permission group required to hide the contained markup.

## 10.19 popover

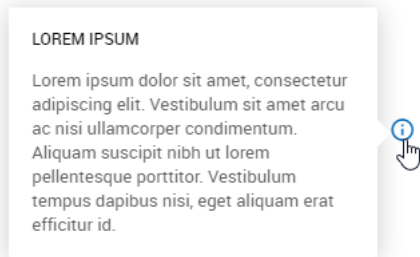
The popover is a one-way binding for enhancing links or buttons with bootstraps [popover](#) functionality.

### 10.19.1 Example

html

```

 <i class="f-20 zmdi zmdi-info-outline"></i>
```



```

```

### 10.19.2 Parameters

- Main parameter

The main parameter can either be a string, which will be used as the text displayed in the popover.

The main parameter can also be an object with the following properties:

- content

The text content to be displayed in the popover.

- title

An optional title for the popover element.

- placement (optional)

Determines where the popover appears. default: auto left

- trigger (optional)

Determines when the popover disappears. default: focus

- html (optional)

Determines if the content is an html. default: false

If one of the parameters is an observable, the binding will update the text, whenever it changes. Additionally if one of the parameters is a resource key, the binding handler will try to get the translated string instead.

## 10.20 requiredPermission

The `requiredPermission` binding causes a section of markup to appear in your document (and to have its `data-bind` attributes applied), only if the current user has a specific permission.

### 10.20.1 Example 1

html

```
<div class="pmb-block pmo-block" data-bind="requiredPermission: { name: 'View', group: 'Tag' }">
 <div class="pmbb-header">
```

```

 <h4>Tags</h4>
 ...
 </div>
 <!-- ko foreach: tags -->
 <button class="btn btn-xs m-5 waves-effect" data-bind="text: Name, click: $parent.toggleTag, css: $parent.getTagButt
 <!-- /ko -->
</div>

```

### 10.20.2 Example 2

Sometimes you may want to control the presence/absence of a section of markup *without* having any container element that can hold an `requiredPermission` binding. For example, you might want to control whether a certain `li` element appears alongside siblings that always appear:

```

`html
 • Add
 • Delete
`

```

### 10.20.3 Parameters

- Main parameter:

The main parameter should be an object with the following properties:

- name  
The permission name required to view the contained markup.
- group  
The permission group required to view the contained markup.

## 10.21 signaturePad

The `signaturePad` binding is used to enhance a `<input type="hidden" />` with the [jQuery SignaturePad plugin](#).

### 10.21.1 Example

```

html
<input type="hidden" data-bind="signaturePad: SignatureJson" />

```

### 10.21.2 Parameters

- Main parameter

The main parameter should be an observable containing the JSON representation of the signature as saved by the signature pad plugin. Whenever the user draws on the canvas, the observable holding the JSON will be updated. If the observable already contains a JSON of a signature, then the *regenerate* function of the SignaturePad API will be called to display the signature.

### 10.21.3 Note

For additional functionality take a look at the `signaturePad` component.

## 10.22 tooltip

The tooltip binding handler initializes a [bootstrap tooltip](#) on an element.

### 10.22.1 Example

```

html
<li data-bind="tooltip: true" data-placement="bottom" title="Filter">
 <a data-trigger="#filter" href="#">

```

```

 <i class="tm-icon zmdi zmdi-search-in-page"></i>


```

### 10.22.2 Parameters

- Main parameter

The main parameter should either be an observable with a value of true or an object containing [options](#) to be passed to the tooltip.

## 10.23 translatedText

The `translatedText` binding translates a given resource key into the current users default language, or to a custom language and sets the text of the element (e.g. `h1`, `p`, `a`, `span`, etc.) to the translated text.

### 10.23.1 Example 1

```

html
<h1 data-bind="translatedText: 'Company'"></h1>

```

### 10.23.2 Example 2: Translating to a specific language

```

html


```

### 10.23.3 Parameters

- Main parameter

The main parameter can either be a string, which will be used as a translation key, or an observable which will be unwrapped and then used as a translation key. If the parameter is an observable, the binding will update the text, whenever the observable changes.

The main parameter can also be an object with the following properties:

- `key`

The translation key as a string or observable (see above)

- `language`

The language in two letter iso language format. This parameter can also either be a string, or an observable. If the parameter is an observable, the binding will update the text, whenever it changes.

- `args`

Optional array of arguments to replace placeholders like `{0}`, `{1}`, etc.

## 10.24 translatedTitle

The `translatedTitle` binding translates a given resource key into the current users default language, or to a custom language and sets the `title` attribute of the element (e.g. `h1`, `p`, `a`, `span`, etc.) to the translated text.

Examples are similar to the `translatedText`.

## 10.25 translatedAriaLabel

The `translatedAriaLabel` binding translates a given resource key into the current users default language, or to a custom language and sets the `aria-label` attribute of the element (e.g. `h1`, `p`, `a`, `span`, etc.) to the translated text.

Examples are similar to the `translatedText`.

## 10.26 translatedDataWmlSpeechCommand

The `translatedDataWmlSpeechCommand` binding translates a given resource key into the current users default language, or to a custom language and sets the `data-wml-speech-command` attribute of the element (e.g. `h1`, `p`, `a`, `span`, etc.) to the translated text.

Use case is only for the Realware HTM-1 glass to enable voice commands.

Examples are similar to the `translatedText`.

## 10.27 userAvatar

The `userAvatar` binding implements the very important functionality to fetch the avatar of a given user and sets it as the `src` of an `<img>` element or the `background-image` css attribute of a `<div>` element. **If you use this binding in a list, especially take care of the parameter given, as it can lead to a lot of database requests.**

### 10.27.1 Example

```
```html
...
```
```

### 10.27.2 Parameters

- Main parameter

The main parameter can be a string of the username or an observable user. If it is the username, the user will be fetched from the database.

### 10.27.3 Note

When the username is not found, the image will not be displayed at all.

When the username is found but the user has no custom avatar, the default avatar will be displayed.

## 10.28 userDisplayName

The `userDisplayName` binding can be used to display the display name for any username. **This binding handler should be used with care, as it will access the database for every binding.**

### 10.28.1 Example 1

```
```html
...
```
```

### 10.28.2 Example 2

```
```html
...
```
```

### 10.28.3 Parameters

- Main parameter

The main parameter can be a string with the username of the user whose display name should be displayed or an object containing the username and an array of users. If the latter is used, the handler will first try to find the user in the provided array and make a query only if the user could not be found.

If the username is an observable value, the binding will update the display name whenever the value changes. If the parameter isn't observable, it will only set the display name once and will not update it again later.

## 10.29 select2autocomplete

The `select2autocompleter` binding extends an `<select>` element with an autocompleter that sets the value of an observable from a selection.

### 10.29.1 Example

```
html
<select class="form-control " name="VisibleToUserIds" multiple="multiple" data-bind="select2autocompleter:
{ data: selectedUsersToDisplay(),autocompleteOptions: { orderBy: ['LastName'], table: 'Main_User',mapDisplayObject: UserMapt
,selectedOptions: selectedUserIds ">
</select>
```

### 10.29.2 Parameters

- Main parameter

When selecting an option from the the autocompleter will be set to the data parameter.

If this parameter is an observable value, the binding will update the element's value whenever the value changes. Example of data : [{id: "german.1", text: "1, German"}, {id: "hungarian.1", text: "1, Hungarian",selected: true}] or [{Value: "german.1", Text: "1, German"}, {Value: "hungarian.1", Text: "1, Hungarian",selected: true}]

- autoCompleteOptions

- table

table is the name of the database table and should be a string. If this field is null an array should be passed as a data parameter.

- orderBy

An array of columns to order by. if it is null the column Name will be used.

- customFilter

An optional parameter with a function which will always be called on the filter query . This function takes the query and term to filter on as parameters. if it is null the function will check if column Name contains the input text. example :  
`function(query, term) { return query.filter(function(x) { return x.FirstName.toLowerCase().contains(this.term) || x.LastName.toLowerCase().contains(this.term); }, { term: term });}`

- mapDisplayObject

function to select the displayed text and value of object found in the database. if it is null the default function will return the id as value and the field name will be displayed. example : `function (objectList) { return $.map(objectList, function (u) { return { id: u.Id, text: u.DisplayName }; });`

};

- onResult

An optional parameter which can be used to do additional work depending on the result of the search. The function is expected to return nothing or a promise.

- onSelect

An optional parameter which can be used to specify a function to call when a suggestion of the autocompleter is selected or when the user sets the input to null.

- placeholder

The placeholder value will be displayed until a selection is made. The default value is "Please select"

- templateResult

An optional parameter with a function which will be used for rendering the results. Can return html or plain text.

- templateResultId

An optional parameter with the id of an element which will be used as a knockout template for rendering the results

- noResults

The text message that will be displayed when no result is found. The default value is "NothingFound"

- onSelect

An optional parameter which can be used to specify a function to call when a suggestion of the autocompleter is selected or when the user sets the input to null.

- joins

In case one table is not enough, the optional joins parameter accepts an array of strings of properties to join.

- confirmChange

An optional function which will be called when the selected element is changed. Can be used to display a confirmation message.

- noneOption

An optional string value. If specified a *None* filter option will be displayed on top of the select list, with the parameter value as id.

- allowClear

An optional boolean parameter to enable or disable the clearing of the select list. The default value is true.

- default An optional default (preselected) value of the select list. If not specified the binding handler will try to resolve it with the value of the data parameter which might require an additional database query.

- nested

An optional parameter that group the results by ParentId, creating a nested structure that will be render in the dropdown. The default value is false

- nestedProperty

A parameter that defines the property that will be use to nest the elements. Is mandatory to use this binding if the nested binding is set to true

## 10.30 text

This binding is an extension to knockout's built-in text binding.

### 10.30.1 Example 1

```
html

```

### 10.30.2 Example 2

```
html

```

### 10.30.3 Example 3

```
html

```

### 10.30.4 Example 4

```
html

```

### 10.30.5 Parameters

- Main parameter

The main parameter is the string to be displayed. If this is the only parameter given the binding will act as the default knockout text binding.

- expand

Controls if the text should be expandable or not. Default value is false.

- limit

Sets the amount of characters to display when expand is set to true. If not set default value is 300 characters.

- whiteSpace

Equals with *white-space* css property. It is *pre-line* by default to keep text formats but any valid values of this css property can be used.

- wordBreak

Equals with *word-break* css property. It is *break-word* by default to avoid UI overflow of long words but any valid values of this css property can be used.

## 10.31 durationInput

The durationInput binding handles the parsing and formatting of an <input> element used for picking durations. Supported formats: hh:mm, hhmm.

### 10.31.1 Example

```
html
<input type="text" class="form-control date-picker" data-toggle="dropdown" aria-expanded="true" data-bind="durationInput: va
```

### 10.31.2 Parameters

- Main parameter

The main parameter should be an observable with a value of either `null`, a `Duration` object or a timespan string. Whenever a duration from the duration picker is selected, the observable will be updated with the `Duration` object representing the selected duration, `null` if the selection was cleared, or the timespan as string when a duration is selected.



## 11 Custom knockout components

Knockout components are a powerful, clean way of organizing your UI code into self-contained, reusable chunks. They were introduced in Knockout v3 and there are already some components you can use.

### 11.1 barcode-format

The barcode-format component displays a preview of a barcode format.

#### 11.1.1 Example

```
```html
```

```
...
```



11.1.2 Parameters

- active

Boolean that determines the bg-color of the component.

- format

String that determines the icon and text content of the component. The options are: - Code-39 - Code-128 - QR-Code

- onClick

11.2 barcode

The barcode component displays a barcode with a specified format.

11.2.1 Example

```
html
<barcode
  params="format: 'CODE39',
          value: 'AdHoc-011040',
          printOnly: true,
          onValidate: (validationResult) => console.log(validationResult)">
</barcode>
```



11.2.2 Parameters

- format

The format of the barcode to be displayed. The options are: - Code-39 - Code-128 - QR-Code

- value

The value of the barcode to be displayed.

- printOnly

An optional boolean parameter that determines whether the barcode will only be displayed in print-mode (css media type: print). By default false.

- onValidate

An optional callback that is invoked after the validation of the barcode value.

Note that the barcode component will display an error alert if the value could not be rendered:

The barcode could not be displayed. The value may be too long or contains invalid characters.

11.3 addressBlock

The addressBlock component formats and displays an *Address* entity.

11.3.1 Example

```
html
<address-block params="address: standardAddress,
                      lookups: lookups,
                      showNames: false">
</address-block>
```

Konica Minolta Business Solutions Europe GmbH
Europaallee 17
30855 Langenhagen
Germany

11.3.2 Parameters

- address

The address object to be displayed.

- lookups

An object with arrays of lookups to be used to display the lookup values of the address. Required by the components are

- regions
- countries
- showCommunicationData

An optional boolean parameter which toggles the display of the Emails, Faxes, Phones and Websites of the address. By default false

- showNames

An optional boolean parameter which toggles the display of the *Name1*, *Name2* and *Name3* properties of the address. By default true

- showLink

An optional boolean parameter which will put the displayed address inside a link to Google Maps. By default true

- currentUserDropboxAddress

Optional parameter what expects the currentUserDropboxAddress ko.computed reference passed from viewModel

11.4 addressEditor

The addressEditor component is a complete editor for an address and the related communication data (*Phones*, *Faxes*, *E-Mails* and *Websites*).

11.4.1 Example

html

```
<address-editor params="companyId: ParentId(),
    contactId: Id(),
    onLoad: $root.onLoadAddressEditor.bind($root),
    editMode: 'person',
    person: person">

</address-editor>
```

Address ^			
Address Type		Work ▲	
Name		Konica Minolta Business Solutions Europe GmbH	
Name 2 e.g. Building or Department			
Name 3 e.g. c/o Name or Room Number			
Street		Europaallee 17	
Zip Code	30855	City	Langenhagen
Zip Code PO Box		PO Box	
Region	Please select ▲	Country	Germany ▲
Phone (1)		▼	
Fax (0)		▼	
E-Mail (0)		▼	
Website (1)		▼	

11.4.2 Parameters

- `onLoad`

An optional parameter which the address editor will call when initializing, passing itself as the only parameter. This can be used to let the address editor show validation errors (`showValidationErrors`) when the user hits the `save` button or ask if the edited address is changes (`isAddressChanged`).

- `companyId`

The contact id, the address is related to.

- `contactId`

The contact id which will be used for new communication data.

- `editMode`

An optional parameter which should be set to *person* when editing the address and communication data related to a person.

- `person`

This parameter is only needed when `editMode` is set to *person* and the person does not yet exist in the database - e.g. when creating a person.

- `addressId`

The id of the address to edit, or null if a new address should be created.

- `lookups`

Optionally, the lookups required by the address editor can be passed as a parameter. If the parameter is omitted the component will read the lookups from the database. The required lookups are:

- AddressType
- Region
- Country
- PhoneType
- FaxType
- EmailType
- WebsiteType
- preFillWithStandardAddress

Optional parameter. Default value: false. If true the standard address of the current company will be selected automatically.

- prepopulatePhone

Optional parameter. Default value: true. If true a new phone entry is initialized automatically when no phone entry exists yet.

- prepopulateFax

Optional parameter. Default value: false. If true a new fax entry is initialized automatically when no fax entry exists yet.

- prepopulateEmail

Optional parameter. Default value: true. If true a new email entry is initialized automatically when no email entry exists yet.

- prepopulateWebsite

Optional parameter. Default value: true. If true a new website entry is initialized automatically when no website entry exists yet.

11.5 addressSelector

The addressSelector component displays a dropdown list for an array of addresses.

11.5.1 Example

```
```html <div class="form-group" data-bind="validationElement: DeliveryAddressId">
```

```
<%= Html.Localize("DeliveryAddress") %>
```

```
<small class="help-block" data-bind="validationMessage: DeliveryAddressId"></small>
```

```
```
```

Delivery Address

Derby Cycle AG, Siemensstrasse 1-3, 49661 Cloppenburg, Germany, (Work)



11.5.2 Parameters

- addresses

An array of addresses to display in the select list

- allowAddNewAddress

Optional parameter which adds an option to create a new address into the select list. This parameter is used by the addressEditor component. By default false.

- addressId

This parameter can be an observable and contains the id of the address that will be preselected. If you are not using onAddressSelect it is expected that addressId is an observable and it will be updated on address selection.

- lookups

The lookups required to format the displayed addresses. The required lookups are:

- AddressType
- Region
- Country
- onAddressSelect

Optionally a callback function can be passed, which will be called with the address id when the selection changes. If `allowAddNewAddress` is true it will pass an empty GUID if the user selects *Add an Address*. If the user selects *Please Select* it will pass null or undefined. `onAddressSelect` allows you to cancel the address selection - e.g. when you want to let the user confirm the change. For this to work you have to return a promise. If you use `onAddressSelect`, it is expected that you update your observables yourself (especially `addressId`).

11.6 barcode scanner

The `barcode-scanner` component displays a button which triggers a barcode scanner. If there is no barcode scanner available then the button is not getting rendered.

11.6.1 Example

```
```html
...

```



### 11.6.2 Parameters

- value

An observable where the barcode scanning result will be written to.

- source

Optional parameter which can be specified together with `column`. If specified then after a barcode was scanned a query on the queryable returned by the function passed as the source will be executed with an additional filter on the column passed as `column`. The result will then be written to the `value` observable instead of the barcode.

- column

Optional parameter, see source.

- id

Optional parameter which can be specified together with `source` and `column`. If specified then the value of the `id` property will be written to the observable instead of the entire entity.

## 11.7 contactData

The `contactData` component displays an address and the related communication data.

### 11.7.1 Example

```
html
<div data-bind="foreach: {data: addresses, as: 'address' }">
 <contact-data params="caption: 'Address #' + ($index() + 1),
 address: address,
 lookups: $root.lookups,
 canDelete: true,
 makeStandardAddress: $root.makeStandardAddress.bind($root),
 loading: $root.loading">
 </contact-data>
</div>
```

## 📞 Contact Data #2 (Invoice-To)

Customer Address	<a href="#">Konica Minolta Business Solutions Austria GmbH</a> <a href="#">Sesamstraße 42</a> <a href="#">1130 Wien</a> <a href="#">Austria</a>
Phone	<a href="#">+43 1 5554711</a> (Front-Desk)
Fax	<a href="#">+43 1 5554712</a> (Work)
E-Mail	<a href="mailto:konicaminolta@example.com">konicaminolta@example.com</a> (Work)
Website	<a href="https://www.konicaminolta.at/de/home.html">https://www.konicaminolta.at/de/home.html</a> (Work)



Edit

Make Standard Address

Delete

### 11.7.2 Parameters

- caption

The caption displayed above the address

- address

The address object to be displayed

- lookups

An object with arrays of the lookups required to display the data, these are:

- addressTypes
- phoneTypes
- faxTypes
- emailTypes
- websiteTypes
- canDelete

An optional boolean parameter whether the address should be deletable. By default `false`

- makeStandardAddress

An optional function parameter which is called when the "Make Standard Address" action is selected. If the parameter is omitted, the action won't be displayed.

- loading

An optional observable parameter which will be updated with the loading state.

- editParameters

An optional string parameter which will be appended to the edit action link.

- phones

Optionally specifies which phones should be displayed. If the parameter is omitted, the *Phones* property of the address will be used.

- faxes

Optionally specifies which faxes should be displayed. If the parameter is omitted, the *Faxes* property of the address will be used.

- emails

Optionally specifies which emails should be displayed. If the parameter is omitted, the *Emails* property of the address will be used.

- websites

Optionally specifies which websites should be displayed. If the parameter is omitted, the *Websites* property of the address will be used.

- contactType

Optionally specifies which contact type this contact data belongs to. Is used for the related permission checks. By default *Company*.

## 11.8 countWidget

The countWidget is a component for displaying counts for each entry when grouping entities.

### 11.8.1 Example

```
html
<count-widget params="caption: 'ServiceCases',
 options: window.Helper.ServiceCase.getCountWidgetOptions">
</count-widget>
```

### 11.8.2 Parameters

- caption

Title of the widget

- query

A JayData queryable which will be used as the source of the displayed data.

- groupBy

Name of the property of which the query will be grouped.

- lookupTable

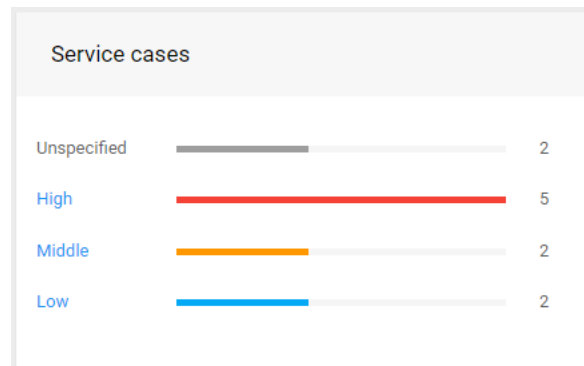
Name of the lookup table (eg. CrmService\_ServicePriority). If defined, the values collected during grouping will be translated from the table.

- urlTemplate

A URL template, which is used when the user clicks on a grouped value. {0} parameter is used to pass the selected value.

- options

an object with the following parameters: query, groupBy, lookupTable, urlTemplate.

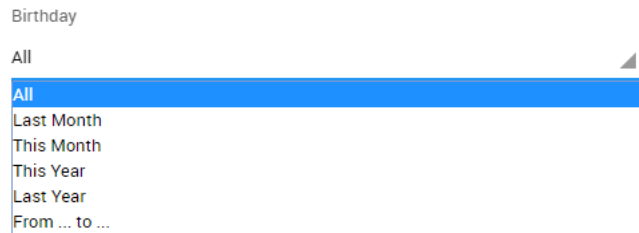


## 11.9 dateFilter

The dateFilter component allows selecting a date range and is used by generic list filters.

### 11.9.1 Example

```
html
<div data-bind="component: {
 name: 'date-filter',
 params: {
 allowFutureDates: false,
 allowPastDates: true,
 caption: 'Birthday',
 value: filter
 }
}"></div>
```



### 11.9.2 Parameters

- `allowFutureDates`

adds future dates to the select list ("Next 3 months", "Next month") and allows selecting future dates from the datepickers shown when selecting "From ... to ..."

- `allowPastDates`

adds past dates to the select list ("Last month", "Last year")

- `caption`

The caption displayed above the select list

- `value`

The observable which will be updated with the selected date / date range

## 11.10 emptyStateBox

The `emptyStateBox` is an emotional component for displaying empty states in any context.

### 11.10.1 Example

```
html
<empty-state-box params="mood: 'happy',
 title: 'ContactTasksEmptyStateTitle',
 text: 'NoTasksLeftInfo'"
 data-bind="visible: tasks().length === 0">
</empty-state-box>
```



## No tasks

Never forget to "Call Peter at 4pm" or "Send the offer to Michael tomorrow morning." You can set a task for yourself or assign one to someone else. The system will even send an email so you don't forget.

### 11.10.2 Parameters

- `mood`

The optional mood parameter supports the three moods *happy*, *sad*, and *excited*. If defined, a L-mon in that mood will appear.

- `title`



The optional title parameter will display a h2 title for the empty state. This parameter can also be a resource key, the component will try to translate it into the current user's language.

- text

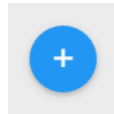
The optional text parameter will display the text of the empty state. This parameter can also be a resource key, the component will try to translate it into the current user's language.

## 11.11 floatingActionButton

The purpose of the `floatingActionButton` component is to represent the primary action of the current context as explained in the [Material design guidelines](#).

### 11.11.1 Example

```
html
<floating-action-button
 data-route="Crm/Note/EditTemplate"
 data-toggle="modal"
 data-target="#modal"
 params="title: 'AddNote'">
</floating-action-button>
```



### 11.11.2 Parameters

- href

The value the href attribute of the action button.

- icon

The icon of the action. By default *plus*. This should be an icon from the [Material Design Iconic Font](#) and will be prefixed with *zmdi-*.

- title

The title of the action. By default *T\_AddNew*. This parameter can also be a resource key, the component will try to translate it into the current user's language.

### 11.11.3 Support for multiple actions

If you want to support multiple actions you need so use a slightly different definition. If you only provide one child `li` element, then the main button will automatically be replaced by the only child. html

```
<floating-action-button params="{ multiButtonMainIcon: 'plus' }">

 <a href="#" data-mfb-label="<%= Html.Localize("AddBusinessRelationship") %>" data-route="Crm/Relationship/Edi
 <i class="zmdi zmdi-accounts-alt"></i>

 <a href="#" data-mfb-label="<%= Html.Localize("AddProjectRelationship") %>" data-route="Crm/Relationship/EditTem
 <i class="zmdi zmdi-file"></i>

</floating-action-button>
```

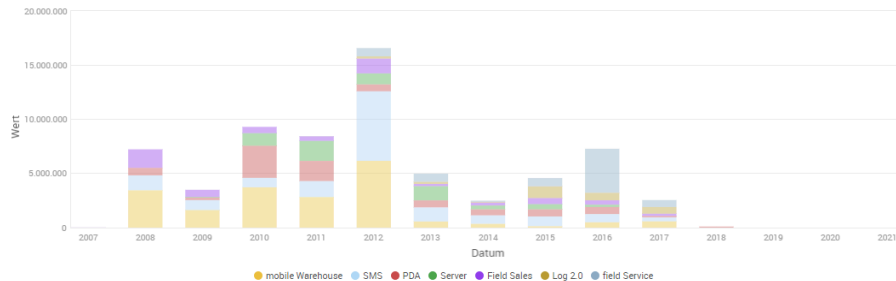
## 11.12 flotChart

The `flotChart` component uses the `flotChart` binding handler to feed a [Flot chart](#).

### 11.12.1 Example

```
html
<flot-chart params="source: database.CrmProject_Project,
 mapAndGroupBy: mapAndGroupByFunc,
 axisXLabel: 'Date',
 axisYLabel: 'Value'">

</flot-chart> js
mapAndGroupByFunc = function(query) {
 return query
 .map(function(it) {
 return {
 d: it.CategoryKey,
 x: (it.DueDate / 1000).strftime("%Y", "unixepoch"),
 y: it.Value.sum()
 };
 })
 .groupBy("it.CategoryKey")
 .groupBy(function(it) { return (it.DueDate / 1000).strftime("%Y", "unixepoch") });
};
```



## 11.12.2 Parameters

- source

A `JayData` queryable which will be used as the source of the displayed data.

- `mapAndGroupBy` A function performing the mapping and the grouping. The mapping is expected to result in an object with the following properties:
  - `d` The label of the result
  - `x` The value on the x-axis
  - `y` The value on the y-axis

For the grouping, usually only two `groupBy` function calls are required, the property of the x-axis and the property of the y-axis.

- `axisXLabel` An optional parameter with the caption of the x-axis. This parameter can also be a resource key, the component will try to translate it into the current user's language.
- `axisYLabel` An optional parameter with the caption of the y-axis. This parameter can also be a resource key, the component will try to translate it into the current user's language.
- `height` An optional parameter to specify a custom height for the chart.
- `unit` An optional parameter to specify the displayed unit of the elements.
- `options` An optional options object which will be passed to the `flot` library, for a list of all supported options please consult the [Flot API documentation](#).

## 11.13 formElement

The purpose of the `formElement` component is to display any type of form element with the correct markup required by bootstrap and the material admin theme.

### 11.13.1 Example

```
html
<form-element params="value: DueDate, caption: 'DueDate', type: 'datePicker'"></form-element>
<form-element params="value: Value, caption: 'Value', type: 'text'"></form-element>
<form-element params="value: SourceTypeKey, caption: 'SourceTypeKey', type: 'select'">
 <select class="form-control" data-bind="value: value, options: $root.sourceTypes, optionsText: 'Value', optionsValue
```

```

 </select>
 </form-element>
 <form-element params="value: BackgroundInfo, caption: 'BackgroundInfo', type: 'textarea'"></form-element>

```

### 11.13.2 Parameters

- **caption**

The caption displayed as the label of the form element. This parameter can also be a resource key, the component will try to translate it into the current user's language.

- **type**

The type of the input which should be rendered, currently supported are:

- email
- password
- select
- text
- textarea
- datePicker
- checkbox
- number
- durationPicker
- timePicker
- fileResource
- select2
- slider
- value

The observable which should be bound to the form element. If the parameter is a validated observable, rule violations will be displayed above the element. Also if the validated observable has a required rule, a required marker will be displayed next to the label.

- **accept**

Optional parameter where accepted file formats can be defined for file inputs.

- **disable**

Optional value or observable which determines if the input should be disabled.

- **placeholder**

Display a placeholder (for keyboard input fields).

- **quantityStep**

Optional parameter which set increasement or decreasement step of number inputs.

- **min**

Optional parameter which set minimum value of number or money inputs.

- **max**

Optional parameter which set maximum value of duration inputs. This value should be a moment duration object.

## 11.14 fullCalendar

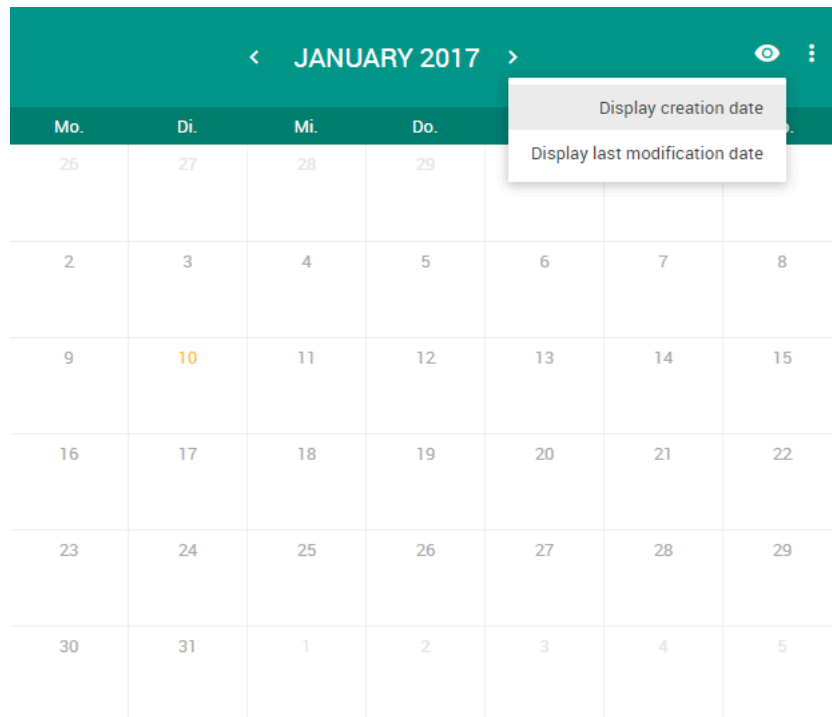
The fullCalendar component uses the fullCalendar binding handler to feed a [FullCalendar](#) widget with an observable array of arbitrary objects. Additionally the component supports rendering custom markup inside the toolbar of the calendar.

### 11.14.1 Example

```
```html <full-calendar params="items: items, getEvent: getTimelineEvent.bind($data), start: timelineStart, end: timelineEnd">
```

- - [Display creation date](#)
 - [Display last modification date](#)
- - [Month](#)
 - [Week](#)
 - [Day](#)

```
</full-calendar> ```
```



11.14.2 Parameters

- items

An array of items which should be displayed in the calendar.

If this parameter is an observableArray, the calendar will update the events whenever the observableArray changes.

- getEvent

A function mapping the passed object to a [FullCalendar Event Object](#).

- editable

Optional parameter which determines whether the events on the calendar can be modified. By default false

- firstHour

Optional parameter which determines the first hour that will be visible in the scroll pane. By default 7

- loading

Optional parameter which can be an observable toggling the loading state of the calendar.

- selectable

Optional parameter which allows a user to highlight multiple days or timeslots by clicking and dragging. By default false

- selectHelper

Optional parameter whether to draw a "placeholder" event while the user is dragging. By default false

- start

This parameter should be used to pass an observable, which will be updated with the first visible day in the calendar whenever the user changes the displayed view. It can be used to trigger lazy loading of displayed elements. Note that in the monthly view this is often before the 1st day of the month, because most months do not begin on the first day-of-week.

- end

This parameter should be used to pass an observable, which will be updated with the last visible day in the calendar whenever the user changes the displayed view. It can be used to trigger lazy loading of displayed elements.

11.15 generic-list-selection

The generic-list-selection adds a search-button which opens a modal containing a generic list which can be filtered and then a result can be selected. It can be rendered next to a dropdown or select2 autocomplete to provide additional filter options.

11.15.1 Example

```
```html
```



```
```
```

11.15.2 Parameters

- value

When selecting an option from the the generic list modal, its id will be set to the observable passed as the value.

- controller

The name of the generic list controller which will be used to render the modal.

- plugin

The name of the plugin of the generic list controller which will be used to render the modal.

- caption

The title of the modal

- disable

An optional parameter which can be a boolean or an observable which conditionally disables the rendered button.

- customFilter

An optional parameter with a function which will always be called on the filter query. The returning query will be the base query used by the generic list.

- onSelect

An optional parameter which can be used to specify a function to call when a search result in the generic list modal is selected.

11.16 inline editor

The inline-editor components allows to have a quick edit of the items in generic lists.

In the item template actions, a new actions must be define in order to show or hide the editor. The setEditing defined in the generic list viewmodel, will receive the viewModel and the item as parameters, to set the isEditing. Which will show or hide the inline editor.

```
html
<li data-bind="if: $parents[2].setEditing">
  <a href="#" data-bind="click: $parents[2].setEditing">
    @(Html.Localize("Edit"))
  </a>
</li>
```

11.16.1 Example

```
```html <div class="inline-editor" data-bind="if: $parent.isEditing && $parent.isEditing() === Id()" > <inline-editor params="context: $data ">
```



```
</inline-editor> ```
```

Potentials

All

M POT000002 - AR Institute

Name \* AR Institute

Source type \* Mailing

Company \* Derby Cycle AG (49661 Cloppenburg, Siemensstrasse 1-3)

Product family Hardware

Priority Please select

Responsible user 1, Default

CANCEL SAVE

C POT000001 - Lifecycle Budget 2021

Lifecycle-Budget L-mobile Sales 2021

Belongs to Derby Cycle AG

Created on: Jun 23, 2021, 3:45:59 PM

Closed: Feb 25, 2022, 11:35:43 AM

Status: Closed A.1, Default 80% (Project budgeted)

### 11.16.2 Parameters

- context

The ko context used inside the component.

- onInit

An optional function that will be called **only once** when the editor is done initializing. The function is called with the editor viewmodel as parameter.

- onBeforeSave

An optional function that will be called when the save button was hit, but before any saving to database begins. This can be used for advanced validations. The function is called with the editor viewmodel as parameter and should return a promise. If the promise is rejected the saving will be stopped.

- onSave

An optional function that will be called right before saving to database. The function is called with the editor viewmodel as parameter. This is the point where you should attach entities to the database if needed. You can return a promise.

- onAfterSave

An optional function that will be called after saving to database is done. The function is called with the editor viewmodel as parameter.

- onCancel

An optional function that will be called when the cancel button was hit. The function is called with the editor viewmodel as parameter.

## 11.17 miniChart

The miniChart component displays a small dashboard widget.

### 11.17.1 Example

```
```html
```

```
![[ko.component.miniChart](img/ko.component.miniChart.png)]

### Parameters
```

```

- `caption`

The caption or the resource key of the caption to be displayed.

- `count`

A function returning a promise resolving with the amount to be displayed.

- `color`

Optional parameter with the background color of the widget.

- `url`

To URL to navigate to when clicking the widget.

## note
The `note` component displays the text of a note. It checks if the note is system generated and will display a
To generate the text, registered providers are used. To register one, call `window.Crm.ViewModels.NoteViewMode
### Example I

```

```
<span data-bind="component: { name: 'note', params: { note: theNote } }">
```

```

### Example II
If the current $data is the actual note, you can just use:

```

```
<span data-bind="component: 'note'"> ``
```

11.18 pmbb

The pmbb component (pmb-block) can be used to display a block inside a details page, which can be toggled between view-mode and edit-mode.

11.18.1 Example

```


html
<pmb-block params="icon: 'account', caption: 'General', context: { project: project }">
  <pmbb-view>
    <!-- ko with: project -->
    <pmbb-view-entry params="caption: 'Name'">
      <span data-bind="text: Name"></span>
      <span class="c-gray" data-bind="translatedText: 'Unspecified', ifnot: Name"></span>
    </pmbb-view-entry>
    <pmbb-view-entry params="caption: 'DueDate'">
      <span data-bind="dateText: DueDate"></span>
      <span class="c-gray" data-bind="translatedText: 'Unspecified', ifnot: DueDate"></span>
    </pmbb-view-entry>
    <pmbb-view-entry params="caption: 'ResponsibleUser'">
      <span data-bind="userDisplayName: ResponsibleUser"></span>
      <span class="c-gray" data-bind="translatedText: 'Unspecified', ifnot: ResponsibleUser"></span>
    </pmbb-view-entry>
    <pmbb-view-entry params="caption: 'SourceTypeKey'">
      <span data-bind="lookupValue: SourceTypeKey, lookups: $root.sourceTypes()"></span>
      <span class="c-gray" data-bind="translatedText: 'Unspecified', ifnot: SourceTypeKey"></span>
    </pmbb-view-entry>
    <!-- /ko -->
  </pmbb-view>
  <pmbb-edit>
    <!-- ko with: project -->
    <pmbb-edit-entry params="caption: 'Name', validationElement: Name">
      <input type="text" class="form-control" data-bind="value: Name">
    </pmbb-edit-entry>
    <pmbb-edit-entry params="caption: 'DueDate', validationElement: DueDate">
      <input type="text" class="form-control date-picker" data-bind="datePicker: DueDate" placeholder="Click here">
    </pmbb-edit-entry>
    <pmbb-edit-entry params="caption: 'ResponsibleUser'">
      <select class="form-control" data-bind="value: ResponsibleUser, options: $root.users, optionsCaption: 'Please select',

```

```

    </select>
  </pmbb-edit-entry>
  <pmbb-edit-entry params="caption: 'SourceTypeKey'">
    <select class="form-control" data-bind="value: SourceTypeKey, options: $root.sourceTypes, optionsText: 'Value', option:
    </select>
  </pmbb-edit-entry>
  <!-- /ko -->
</pmbb-edit>
</pmb-block>

```

 General

⋮

Name

Bearbeitungszentrum CNC 1200

Edit

Due Date


Oct 31, 2011

Responsible User

1, Default

Source type

Onlinemarketing

 General

⋮

Name

Bearbeitungszentrum CNC 1200

Due Date

Oct 31, 2011

📅

Responsible User

1, Default

▾

Source type

Onlinemarketing

▾

SAVE

CANCEL

11.18.2 Parameters

The `<pmb-block>` itself accepts the following parameters:

- `icon`

The icon of the action. By default *plus*. This should be an icon from the [Material Design Iconic Font](#) and will be prefixed with *zmdi-*.

- `caption`

The caption displayed on top of the block. This parameter can also be a resource key, the component will try to translate it into the current user's language.

- `context`

The `ko` context used inside the component. Can be a single observable, or an array of observables.

- `css`

Optional additional css class(es) for the block.

- `condition`

An optional boolean condition whether the block can be switched into edit mode. If this parameter is an observable value, the binding will update the element's value whenever the value changes. If the parameter isn't observable, it will only set the element's value once and will not update it again later. By default the edit mode can always be enabled.

- `onInit`

An optional function that will be called **only once** when the block is done initializing. The function is called with the block's viewmodel as parameter.

- `onBeforeSave`

An optional function that will be called when the save button was hit, but before any saving to database begins. This can be used for advanced validations. The function is called with the block's viewmodel as parameter and should return a promise. If the promise is rejected the saving will be stopped.

- `onSave`

An optional function that will be called right before saving to database. The function is called with the block's viewmodel as parameter. This is the point where you should attach entities to the database if needed. You can return a promise.

- `onAfterSave`

An optional function that will be called after saving to database is done. The function is called with the block's viewmodel as parameter.

- `onCancel`

An optional function that will be called when the cancel button was hit. The function is called with the block's viewmodel as parameter.

- `showHeader`

An optional boolean condition whether the header of the block should be displayed. If this parameter is an observable value, the binding will update the element's visibility whenever the value changes. If the parameter isn't observable, it will only set the element's visibility once and will not update it again later. By default the header will always be shown.

- `buttons`

An optional array, each element should contain an object with the following properties: 1. `css`: optional string 2. `text`: optional string 3. `click`: optional function 4. `requiredPermission`: optional object

These buttons will be rendered under the Edit button of the `pmbb-block`.

The `<pmbb-view-entry>` elements inside the `<pmbb-view>` container element accept the following parameters:

- `caption`

The caption of the entry. This parameter can also be a resource key, the component will try to translate it into the current user's language.

The `<pmbb-edit-entry>` elements inside the `<pmbb-edit>` container element accept the following parameters:

- `caption`

The caption of the entry. This parameter can also be a resource key, the component will try to translate it into the current user's language.

- `validationElement`

An optional parameter which will be passed to the `validationElement` binding of the entry.

Please note: validations on `ExtensionValues` only works when context set as follows: ``html

```
<pmbb-edit-entry id="InitiatorGroup" params="caption: 'CustomerInitiatorGroups', validationElement: InitiatorGr
  <select class="form-control" data-bind="select2autocompleter: { data: InitiatorGroupKey, autocompleteOption
  </select>
</pmbb-edit-entry>
```

...

11.19 scaleFilter

The `scaleFilter` component allows selecting predefined values (ex. `>100`) from dropdown (depending on the given min, max, step and operator values). Additionally there is an option for Custom, where the user can add exact min and max values. Used by generic list filters.

11.19.1 Example

```
html
<div data-bind="component: {
  name: 'scale-filter',
  params: {
    min: MinValue(),
    max: MaxValue(),
    step: Step(),
    caption: $root.getLocalizationText($data),
    value: Response,
    ignoreCaption: true
  }
}"></div>
```

11.19.2 Parameters

- min

Defines the min value of the filter.

- max

Defines the max value of the filter.

- step

Defines the step value of the filter.

- caption

Defines the caption key of the filter. By default the translation for the key will be the caption of the filter.

- value

The observable used to save the value of the filter. This is the value passed to the `formElement` component.

- ignoreCaption

Boolean parameter if true then the caption key will be used as it is for the filter caption (useful if the filter is for custom named properties ex. attribute form), if false then the caption will be the translated caption key.

11.20 signaturePad

The `signaturePad` component is based on the `signaturePad` binding handler used to bind an observable to the input of a [jQuery SignaturePad plugin widget](#). Unlike the binding handler, this component can not only save the JSON representation of the signature, but also the name of the signee and the date the signature was done.

11.20.1 Example

```
html
<signature-pad params="signature: Signature,
  signatureDate: SignatureDate,
  signatureName: SignatureName">
</signature-pad>
```

Signature

Peter Pan



RESET SIGNATURE

11.20.2 Parameters

- signature

The observable used to save the JSON representation of the signature to, or regenerate an existing signature from. This is the value passed to the signaturePad binding handler.

- signatureDate

The observable which will be updated with the current date as soon as the signature is done. When the signature is reset, the value of the observable is set to null.

- signatureName

The observable which will be updated with the name entered in the text input above the signature pad.

11.21 timeRangeFilter

The timeRangeFilter component allows selecting a time range and is used by generic list filters.

11.21.1 Example

```
html
<div data-bind="id:'formelement-' + Id() ,component: {
  name: 'timeRange-filter',
  params: {
    caption: $root.getLocalizationText($data),
    value: Response,
    ignoreCaption: true
  }
}"></div>
```

11.21.2 Parameters

- caption

Defines the caption key of the filter. By default the translation for the key will be the caption of the filter.

- value

The observable used to save the value of the filter. This is the value passed to the timePicker.

- ignoreCaption

Boolean parameter if true then the caption key will be used as it is for the filter caption (useful if the filter is for custom named properties ex. attribute form), if false then the caption will be the translated caption key.

11.22 taskListBlock

Generic list of tasks in a block component.

11.22.1 Example

```
html
<task-list-block params="parentViewModel: ParentViewModel">
</task-list-block>
```

11.22.2 Parameters

- parentViewModel

The parent view model which has to be a ContactDetailsViewModel descendant. Mandatory.

11.23 statusChooser

This component is responsible for the status change.

11.23.1 Example

```
html
<status-chooser params="currentStatus: CurrentStatus,
                                statuses: Statuses,
                                settableStatuses: SettableStatuses,
                                setStatus: SetStatus,
                                canSetStatus: CanSetStatus"></status-chooser>
```

11.23.2 Parameters

- `currentStatus`

The key of the current status. Mandatory.

- `statuses`

An array of all statuses. Elements for which the key is not specified should be excluded. Mandatory.

- `settableStatuses`

An array of currently settable statuses. Mandatory.

- `setStatus`

A delegate of a function that handles the status changes. When the status change action is route based, use parameter `setStatusRoute` instead otherwise this parameter is mandatory.

- `setStatusRoute`

The action route as string. It is only mandatory when there's no status setter action (parameter `setStatus` is not set).

- `canSetStatus`

A delegate of a function that determines whether the status can be set. Its return value is Boolean. By default the component is disabled.

11.24 collapsibleBlock

A collapsible container component. It consists of two sub components: a header and a content section. The latter can be collapsed/expanded.

11.24.1 Example

```
html
<collapsible-block params="id: Id, onLoad: $root.onLoadCollapsibleBlock.bind($root)">
  <collapsible-block-header params="icon: Icon">
  </collapsible-block-header>
  <collapsible-block-content>
  </collapsible-block-content>
</collapsible-block>
```

11.24.2 Parameters

11.24.2.1 collapsible-block

- `id` A string id, must be unique inside a given view. Mandatory.
- `collapsed` A boolean that specifies whether the content section should be collapsed or not by default. The default value is `false`.
- `onLoad` An optional parameter which the collapsible block will call when initializing, passing itself as the only parameter.

11.24.2.2 collapsible-block-header

- `customCss` Additional css class(es) as a string can be set with this paramter. These will be applied to the `card-header` div.
- `icon` The name of the icon. It is undefined by default. This should be an icon from the [Material Design Iconic Font](#) and will be prefixed with `zmdi-`.
- `swapIcons` A booealan that specifies whether the icons (collapse/expand chevron and the main icon) should be swapped. By default the collapse/expand chevron is displayed on the right.
- `context` An arbitrary object that acts as the context of the content.

collapsible-block-content

- `context` An arbitrary object that acts as the context of the content.

11.25 block

A simple container component. It consists of two sub components: a header and a content section.

11.25.1 Example

```
html
<block>
  <block-header>
</block-header>
  <block-content>
</block-content>
</block>
```

11.25.2 Parameters

11.25.2.1 block-header

- `customCss` Additional css class(es) as a string can be set with this paramter. These will be applied to the `card-header` div.
- `icon` The name of the icon. It is undefined by default. This should be an icon from the [Material Design Iconic Font](#) and will be prefixed with `zmdi-`.
- `context` An arbitrary object that acts as the context of the content.

11.25.2.2 block-content

- `context` An arbitrary object that acts as the context of the content.

12 Custom knockout extenders

If you wish to add additional functionality to an observable (ex. different read/write behaviours) use extenders. There are already some defined in the application.

12.1 filterOperator

The `filterOperator` extender used at the rendering of filters in material client. This extender is responsible for the javascript generic list filter's structure.

12.1.1 Example

Example for single parameter usage

```
html
<% if (Model.Item.FilterType == typeof(bool)) { %>
  <%var modelPath = Model.Item.GetModelPath();
  var read = $"function() {{ var value = ko.unwrap(this.getFilter('{modelPath}')); return value !== null && value !== unde
  var write = $"function(newValue) {{ this.getFilter('{modelPath}').extend({{ filterOperator: {'==='} }})(newValue); }}";
  var data = $"ko.pureComputed({{ read: {read}, write: {write} }}, $data)"; %>
  <div class="fg-line select">
    <p class="m-b-5"><%= Html.Localize(caption) %></p>
    <select name="<%= Model.Item.GetModelPath() %>" data-bind="value: <%= data %>, options: booleanOptions, optionsValue:
    </select>
  </div>
<% }
```

Example for single parameter usage - with operator object

```
html
<% if (Model.Item.FilterType == typeof(bool)) { %>
  <%var modelPath = Model.Item.GetModelPath();
  var read = $"function() {{ var value = ko.unwrap(this.getFilter('{modelPath}')); return value !== null && value !== unde
  var write = $"function(newValue) {{ this.getFilter('{modelPath}').extend({{ filterOperator: {{ type: 'some', collectionNi
  var data = $"ko.pureComputed({{ read: {read}, write: {write} }}, $data)"; %>
  <div class="fg-line select">
    <p class="m-b-5"><%= Html.Localize(caption) %></p>
    <select name="<%= Model.Item.GetModelPath() %>" data-bind="value: <%= data %>, options: booleanOptions, optionsValue:
    </select>
  </div>
<% }
```

Example for additional parameter usage

```
html
<% if (Model.Item.FilterType == typeof(bool)) { %>
  <%var modelPath = Model.Item.GetModelPath();
  var read = $"function() {{ var value = ko.unwrap(this.getFilter('{modelPath}')); return value !== null && value !== unde
  var write = $"function(newValue) {{ this.getFilter('{modelPath}').extend({{ filterOperator: {{ operator: '===', addition
  var data = $"ko.pureComputed({{ read: {read}, write: {write} }}, $data)"; %>
  <div class="fg-line select">
    <p class="m-b-5"><%= Html.Localize(caption) %></p>
    <select name="<%= Model.Item.GetModelPath() %>" data-bind="value: <%= data %>, options: booleanOptions, optionsValue:
    </select>
  </div>
<% }
```

12.1.2 Parameters

- operator

This parameter will define the filter's Operator attribute. The first two example above shows how to add a simple operator value or add an operator object.

- additionalProperties

The parameters defined in `additionalProperties` object will be added as properties to the javascript filter object. In this case you need to call the extender with an object what has an `additionalProperties` named object as property. In this inner object you can define the properties what you wanna use in javascript filters.\ See third example for usage.

13 Recipes for developing in Customer Plugins

The following document will give you an insight into often used functions in the customization of the Crm / Service application. Please use it as a reference to get guidelines for developing new or changing existing features. If you find the information to be outdated or inadequate please contact either [Oliver Joest](#) or [Oliver Titze](#).

13.1 Working with Customer Plugins

As the system grows and expands we keep all code for a single customer at a central location, the **Customer Plugin**. A running system is designed to stack the plugins the right way, so that a customer plugin may expand or override the underlying base system.

You can think of the system as a layered cake with the following structure

Customer Plugin Layer

Generic Plugin Layer

Base layer

Whereas the customer Plugin is the highest available layer today. To create a customer Plugin some conventions are demanded:

- Name your Customer plugin in the format Customer.[CustomerName]
- The Project should be added to the solution folder *Customer plugins*
- A solution should **never** contain more than **one** individual customer plugin

Example

```
src
  Crm.Web
    Plugins
      Crm.Article
      Crm.Campaigns
      ...
      Customer.CustomerName
      ...
```

- Whenever you choose to add functions you are requested to do so via the Customer Plugin layer
- Whenever you are requested to extend existing base layer functions you are requested to do so via the Customer Plugin

13.1.1 The CustomerNamePlugin.cs file

The customer project will contain a file called [CustomerName]Plugin.cs. This file will include the following elements

```
namespace Customer.ACME
{
    using Crm.Library.Modularization;

    [Plugin(Requires="Crm.Article,Crm.Service")]
    public class CustomerACMEPlugin : CustomerPlugin
    {
        ...
    }
}
```

In brief the following file contains several important informations.

13.1.2 PluginAttribute

The class Attribute *PluginAttribute* allows you to specify other required Plugins that the customer plugin requires. The order specified does not matter for the dependency selection.

Although it is not necessary to type all direct and indirect dependencies of the plugin, it is considered best practice to include all dependencies for a better overview. E.g. Crm.Service already requires Crm.Article and could be omitted from the customer plugin

13.1.3 CustomerPlugin inheritance

The class itself should inherit the class *CustomerPlugin* and not the more generic *Plugin* class as this would leave the system unable to sort the Customer plugin to the highest level of layering.

13.2 Background services

Systems often need to carry out time consuming tasks that may be run asynchronously. Sometimes systems should just execute tasks in certain intervals (e.g. every minute) to process other data. Either way you will want to create some infrastructure to carry out recurring tasks. These are called Background services and come in 2 flavors.

13.2.1 Background Services - Automatic Session Handling

When your customer project demands for recurring processes to be executed you can include new Background Services in the corresponding folder:

```
Customer.ACME
  BackgroundServices
    CustomBackgroundService.cs
  jobs.xml
```

The class will contain at least the following elements

```
namespace Customer.ACME.BackgroundServices
{
    using Crm.Library.BackgroundServices;
    using Quartz;

    [DisallowConcurrentExecution]
    public class CustomBackgroundService : JobBase
    {
        protected override void Run(IJobExecutionContext context)
        {
            // Periodic executed Code goes here
        }
    }
}
```

Jobs that are created this way will automatically handle session transactions before and after the Run method. Please be aware that you should not execute long running tasks in this context, because holding the transaction open for a longer time will lead to deadlocks on the tables. If you need to manage the session and transaction times your self you may choose

13.2.2 Background Services - Manual Session Handling

With manual session handling you have the chance to open and close transactions while you process your data. This is useful if you have plenty of records that need processing in batches. By using the manual session handling you're able to control the locking time of incorporated tables. If there are long running calls against external systems you should choose the Manual Session Handling.

```
namespace Customer.Grimme.BackgroundServices
{
    using System;
    using System.Linq;

    using Crm.Library.BackgroundServices;
    using Crm.Service.Model;

    using Customer.Grimme.Services.Interfaces;

    using Quartz;

    [DisallowConcurrentExecution]
    public class OrderExportAgent : ManualSessionHandlingJobBase
    {
        protected override void Run(IJobExecutionContext context)
        {
            var collection = exportService.GetUnexportedEntities().ToList();

            foreach (var entity in collection)
```

```

    {
        try
        {
            BeginTransaction();
            exportService.Export(entity);
            EndTransaction();
        }
        catch (Exception e)
        {
            Logger.Error(String.Format("Error exporting entity {0}", entity.Id), e);
            RollbackTransaction();
        }
    }
}
}

```

As you can see above an individual session will be created for each order exported.

Bottom line: Choose the manual session handling if you need full control, choose the job base if the automatic opening and closing of sessions is sufficient

13.2.3 Background Services - jobs.xml

The jobs.xml file in the root of the customer plugin will specify a trigger for the new background processing task. This will look like

```

<job-scheduling-data xmlns="http://quartznet.sourceforge.net/JobSchedulingData"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="2.0">

  <processing-directives>
    <overwrite-existing-data>true</overwrite-existing-data>
  </processing-directives>

  <schedule>
    <job>
      <name>CustomBackgroundService</name>
      <group>Customer.ACME</group>
      <description>Executed periodically.</description>
      <job-type>
        Customer.ACME.BackgroundServices.CustomBackgroundService,
        Customer.ACME
      </job-type>
      <durable>true</durable>
      <recover>false</recover>
    </job>
    <trigger>
      <cron>
        <name>CustomBackgroundServiceTrigger</name>
        <group>Customer.ACME</group>
        <description>Runs every five minutes.</description>
        <job-name>CustomBackgroundService</job-name>
        <job-group>Customer.ACME</job-group>
        <cron-expression>0 0/5 * 1/1 * ?</cron-expression>
      </cron>
    </trigger>
  </schedule>

```

Please consider adding a jobs.template.xml and corresponding jobs.debug.xml and jobs.release.xml files if you wish to transform the contents of the jobs file based on the configuration you're going to build

13.3 Business Rules

Business rules are used to handle form processing in the system. Whenever a form element is rendered using the HtmlHelper extensions the business rules will be attached to the corresponding input field. When the user submits a form to save an entity all active business rules are evaluated and validation messages will be rendered to the failing input element.

13.3.1 CrmModelBinder for new created classes

13.3.2 Overriding rules from Customer plugin

13.3.3 Ignoring rules from Customer plugin

13.3.4 Creating new rules from Customer plugin

Rule Types

- Crm.Library.Validation.BaseRules

13.3.5 HtmlHelperExtensions for Business rules

13.3.5.1 WrappedXXX for server side forms

13.3.5.2 TemplateInput for Knockout bound forms

13.4 Routing

13.4.1 Create new Routes for your customer plugin

If you need any new controllers try to stick to the default route scheme Plugin/Controller/Action/Id. Nonetheless you have to register 2 common routes for your customer plugin:

```
namespace Customer.ACME
{
    using System;

    using Crm.Library.Modularization.Registrars;
    using Crm.Library.Routing.Constraints;

    public class Routes : IRouteRegistrar
    {
        public virtual RoutePriority Priority
        {
            get { return RoutePriority.Low; }
        }
        public virtual void RegisterRoutes(IEndpointRouteBuilder endpoints)
        {
            endpoints.MapControllerRoute(
                null,
                "Customer.ACME/{controller}/{action}/{id?}",
                new { action = "Index", plugin = "Customer.ACME" },
                new { plugin = "Customer.ACME" }
            );
        }
    }
}
```

Please don't register new custom routes because that will impact the performance of the routing module.

13.4.2 Redirecting existing routes to customer plugin

```
public virtual void RegisterRoutes(IEndpointRouteBuilder endpoints)
{
    ...
    endpoints.MapControllerRoute(
        null,
        "Crm.Project/ProjectList/GetAll",
        new
        {
            plugin = "Customer.ACME",
            controller = "ProjectList",
        }
    );
}
```

```
        action = "GetAll"
    }
    );
    ...
}
```

Optionally you can declare different controller and action names if necessary, just specify them in the override definition for Controller and Action parameters e.g.

```
public virtual void RegisterRoutes(IEndpointRouteBuilder endpoints)
{
    ...
    endpoints.MapControllerRoute(
        null,
        "Crm.Project/ProjectList/GetAll",
        new
        {
            plugin = "Customer.ACME",
            controller = "CustomProjectList",
            action = "CustomGetAll"
        }
    );
    ...
}
```

13.5 Menu extensions~~~~

You're able to extend and modify the existing main menus of the application. By registering new menu entries to the left, right or mobile menu you're able to extend the application with custom implementations. To change the default menu structure you have to create a custom `MenuRegistrar` class. To extend certain parts of the menu you may choose to inherit the following interfaces:

- IMenuRegistrar for the menu in the material client
- IMenuRegistrar for the collapsible menu in the material client ~~~~ ``c# namespace Customer.ACME { using Crm; using Crm.Library.Modularization.Menu;

```
public class ACMEMenuRegistrar : IMenuRegistrar, IMenuRegistrar { public virtual void Initialize(MenuProvider
menuProvider) { // operations on menu for material client } public virtual void Initialize(MenuProvider menuProvider) { //
operations on collapsible menu for material client } } ``
```

13.5.1 Unregistering existing menu entries

When you do want to completely remove already existing menu entries from other plugins you need to know the category and title they were registered with. To find out about the plugin you may inspect the incorporated link of the Menu entry e.g. `Auftrag`. With this information you can look up the registered menu entries from Crm.Order plugin by looking at the corresponding OrderMenuRegistrar class. With this information you may:

```
c#
...
public virtual void Initialize(MenuProvider<MainMenu> menuProvider)
{
    ...
    menuProvider.Unregister(category: "Main", title:"Order");
    ...
}
...
```

to remove the main entry completely.

13.5.2 Registering new menu entries

To add new entries you can call the `register` method on your `menuProvider` instance.

```
c#
...
public virtual void Initialize(MenuProvider<MainMenu> menuProvider)
{
    ...
    menuProvider.Register(category: "Main",
                           title: "ACME");
}
```

```

        menuProvider.Register(category: "ACME",
                                title: "CustomProcess",
                                url: "~/Customer.ACME/CustomProcess",
                                priority: 100);
        ...
    }
    ...

```

in the upper example some details are noteworthy. The registration is a 2 step process. In the first step a new category *ACME* is registered onto the *Main* level of the left menu. In the next step a *CustomProcess* is registered in the *ACME* category which leads to the specified Url. If you would need to register multiple entries below the *ACME* category you could customize the order by specifying different priority values.

```

c#
...
menuProvider.Register(category: "ACME",
                        title: "CustomProcess1",
                        url: "~/Customer.ACME/CustomProcess1",
                        priority: 100);

menuProvider.Register(category: "ACME",
                        title: "CustomProcess2",
                        url: "~/Customer.ACME/CustomProcess2",
                        priority: 90);

...

```

When rendering menu entries the category and title attributes will be processed by a localize call. So please make sure to add the corresponding Resource entries for localization.

13.5.3 Menu entries, adding permissions

After you register your new menu entries you can request individual Permissions to access the elements.

```

c#
...
menuProvider.AddPermission(category: "ACME",
                            title: "CustomProcess1",
                            permissionGroup: "ACME",
                            permissionName: "CustomProcess1");
...

```

Please remember to add the corresponding permissions from code or via Database migration afterwards.

13.5.4 Menu entries, adding icons

When registering new menu entries for the material client, the optional named parameter *iconClass* can be used to specify a matching icon. The icon font included in the client is the [Material Design Iconic Font](#).

```

c#
...
menuProvider.AddPermission(category: "ACME",
                            title: "CustomProcess1",
                            iconClass: "zmdi zmdi-home",
                            permissionGroup: "ACME",
                            permissionName: "CustomProcess1");
...

```

13.6 Generating IDs on the client-side

How to generate IDs on the client-side without ID collisions.

13.6.1 Client Side Additions

13.6.1.1 1 - Register the ID provider:

```

window.Crm.Offline.IdProvider.registerHighLowProvider(
    'CrmService_ServiceOrderDispatch',
    'ServiceOrderDispatch',
    'Crm.Service');

```

13.6.1.2 2- Initialize the ID providers

```

window.Crm.Offline.Bootstrapper.initializeIdProviders.then(function() { //you are ready to generate IDs at this

```

13.6.1.3 3- Calculate and attach the new ID when saving your new entities

```

var myNewId = window.Crm.Offline.IdProviders.CrmService_ServiceOrderDispatch.getId();

```

13.6.2 Server Side Changes

- Change class to be EntityBase instead of int
- Migrate table (e.g.
src\Crm.Web\Plugins\Sms.TimeManagement\Database\20140206125023_ChangeSmsTimeManagementEventToHighLowIds.cs)

```

[Migration(20140206125023)]
public class ChangeSmsTimeManagementEventToHighLowIds : Migration
{
    private const int Low = 32;

    public override void Up()
    {
        Database.RemovePrimaryKey("SMS", "TimeManagementEvent");
        Database.RenameColumn("[SMS].[TimeManagementEvent]", "TimeManagementEventId", "Id_Identity");
        Database.ExecuteNonQuery("ALTER TABLE [SMS].[TimeManagementEvent] ADD TimeManagementEventId bigint");
        Database.ExecuteNonQuery("UPDATE [SMS].[TimeManagementEvent] SET TimeManagementEventId = Id_Identity");
        Database.ExecuteNonQuery("ALTER TABLE [SMS].[TimeManagementEvent]
            ALTER COLUMN TimeManagementEventId bigint NOT NULL");
        Database.ExecuteNonQuery("ALTER TABLE [SMS].[TimeManagementEvent] DROP COLUMN Id_Identity");
        Database.ExecuteNonQuery("ALTER TABLE [SMS].[TimeManagementEvent] ADD CONSTRAINT PK_TimeManagementEvent
            PRIMARY KEY(TimeManagementEventId)");
        Database.ExecuteNonQuery("BEGIN IF ((SELECT COUNT(*) FROM dbo.hibernate_unique_key
            WHERE tablename = '[SMS].[TimeManagementEvent]') = 0) INSERT INTO hibernate_unique_key
            (next_hi, tablename) values ((select (COALESCE(max(TimeManagementEventId), 0) / " + Low + ")
            from [SMS].[TimeManagementEvent] where TimeManagementEventId is not null), '[SMS].[TimeManagementEvent]')");
    }
    public override void Down()
    {
        throw new NotImplementedException();
    }
}

```

- Change NHibernate key mapping to Generators.Assigned
- Create HighLowDefinition for class

```

public class TimeManagementEventHighLowDefinition : HighLowDefinition
{
    protected override string Tablename
    {
        get { return "[SMS].[TimeManagementEvent]"; }
    }
}

```

- Implement IHighLowEntity<>

```

public class TimeManagementEvent : EntityBase<long>, IHighLowEntity<TimeManagementEventHighLowDefinition>

```

- Change SyncService / Reference to Repositories like:
 - DefaultSyncService -> DefaultSyncService<TimeManagementEvent, long>
 - IRepository -> IRepositoryWithTypedId<TimeManagementEvent, long>

13.7 Creating and querying a hybrid model

This page explains creating an offline (client-side) model and using this model to query both online and offline sources, in a step-by-step fashion.

[[warning:This page is a work in progress]]

13.7.1 Step 1 - Tell the application to generate a client side model for your entity

```
[ProvidesClientSideModelDefinition(typeof(Model.Order), typeof(Model.OrderItem))]  
public class OrderPlugin : Plugin  
{  
    public new static string Name = "Crm.Order";  
}
```

In this example, the application will (try to) generate a client side definition of the entities "Order" and "OrderItem". It is important to add this attribute to the Plugin definition as the generator (GetDefinitions method of `Crm.Offline.Controllers.OfflineController`) will only check those to gather type information. When making queries from `JayData`, the library will use these models to make request on both online and offline databases/services. It is also required that the Entities defined here have a Rest model.

If you want to define your model manually, here's an example to get you started (No need to do this when you use the attributes in your plugin definition):

```
//TODO: to reduce the cargo-cult effect, explain what each step does.  
var definition = {  
    ItemNo: { type: "string" },  
    Description: { type: "string" },  
    ArticleType: { type: "string" },  
    Price: { type: "number" },  
    IsActive: { type: "bool" }  
}  
var modelPath = "Crm.Article.Article"; //by convention  
var modelDbPath = "CrmArticle_Article"; //by convention  
$data.define(modelPath, definition);  
namespace('Crm.Offline').SyncObjects.push(function () { return window.ko.observableArray([]).config({  
    storage: modelDbPath,  
    model: "Article",  
    pluginName: "Crm.Article"  
});  
});  
window.Helper.Offline.registerTable("CrmArticle_Article", "Article", "Crm.Article", definition);  
Offline.EntityStore = $data.EntityContext.extend("Offline.EntityStore", { "CrmArticle_Article" : definition });
```

13.7.2 Step 2 - Create a sync service

Derive from the `DefaultSyncService<TEntity, TId>`. Application will use this service to decide what to sync and how to save. Here's an example:

```
public class OrderSyncService : DefaultSyncService<Order, long>  
{  
    public override Order Save(Order entity)  
    {  
        /* save the entity here */  
        return entity;  
    }  
    public override IQueryable<Order> GetAll(DateTime? lastSync)  
    {  
        /* entities is available to query the database */  
        return entities.Since(lastSync);  
    }  
    public override IQueryable<Order> Filter(IQueryable<Order> entities, User user, DateTime? lastSync)  
    {  
        return entities.Where(o => o.ResponsibleUser == user.Id).Since(lastSync);  
    }  
}
```

The code is mostly self-explanatory. You can look at the `DefaultSyncService` implementation to have a better idea of what you can override to alter the sync process. Now, when you visit `/Crm.Offline/Offline/Sync`, you'll have your entities synced to the client.

The name of the database will be {plugin name without dots}_{entity name}. For example, Orders will be stored in the CrmOrder_OrderItem table and Articles will be in CrmArticle_Article.

13.7.3 Step 3 - Altering your entity to support client-side ID generation

It's ~~easy~~ *not hard* to allow the creation of new entities on the client side and having the client deal with the new IDs it generates. This step is explained in a separate page: [\[\[link:/project/advanced-topics/targeting-both-offline-and-online-modes/generating-ids-on-the-client-side/|Generating IDs on the client-side\]\]](/project/advanced-topics/targeting-both-offline-and-online-modes/generating-ids-on-the-client-side/)

13.7.4 Step 4 - Using the generated model to query offline and online sources

JayData is used in the Crm to query the data. Here is an example query which will work both online and offline (after a sync):

```
window.currentDatabase.Crm_Company
    .find(companyId)
    .then(function (company) {
        // company is the entity with the given id
    });
```


14 Customizing the ServiceOrderDispatch report

14.1 Overriding & extending the ViewModel

The dispatch report uses the same view for the report preview in the mobile service client and when generating a PDF on the server, but uses two different viewmodels. This means that when changing the logic, or adding additional data, these changes have to be done in both of the viewmodels.

14.1.1 JavaScript ViewModel

The JavaScript viewmodel, which is used to render the dispatch report preview, can be changed by overriding the prototype of the viewmodel. The dispatch report preview uses the same viewmodel as the other tabs in the dispatch details, so the viewmodel located at `window.Crm.Service.ViewModels.DispatchDetailsViewModel` needs to be extended.

14.2 How to define the margins and header / footer sizes

For headers and footer margins, the `DispatchReportViewModel` contains 4 properties:

- `FooterContentSize` (default: 2.5 cm, appsetting: *Report/FooterHeight*)
- `FooterContentSpacing` (default: 0.5 cm, appsetting: *Report/FooterSpacing*)
- `HeaderContentSize` (default: 3.5 cm, appsetting: *Report/HeaderHeight*)
- `HeaderContentSpacing` (default: 0.5 cm, appsetting: *Report/HeaderSpacing*)

To change the default values this can be done by overriding the properties in a custom viewmodel, or by changing the default values in the `appsettings.config` of the Main plugin.

The left and right page margins of the PDF will default to 2.5 cm and 2 cm and can be further modified using CSS margins.

14.3 Add & format paging indicators

The dispatch report headers and footers are added to the PDF using iTextSharp PDF stamping. For each page the parameters `[page]` and `[toPage]` will be replaced before rendering to display the current page and the total number of pages:

```
html
Page <span>[page]</span> of <span>[toPage]</span>
```

14.4 Include additional CSS & JS assets

To include additional CSS or JS assets, the `PluginRenderAction TemplateHeadResource` can be used:

```
c#
[AllowAnonymous]
[RenderAction("TemplateHeadResource", Priority = 50)]
public virtual ActionResult TemplateReportCss()
{
    return Content(WebExtensions.CssLink("templateReportCss"));
}
[AllowAnonymous]
[RenderAction("TemplateHeadResource", Priority = 50)]
public virtual ActionResult TemplateReportJs()
{
    return Content(WebExtensions.JsLink("templateReportJs"));
}
```

14.5 Stamping and appendices

To stamp files (e.g. the company stationery) the *Stamp* method of the `IPdfService` can be used:

```
c#
public class CustomServiceOrderService : ServiceOrderService, IServiceOrderService, IReplaceRegistration<ServiceOrderService>
{
    public override byte[] CreateDispatchReportAsPdf(ServiceOrderDispatch dispatch)
```

```

    {
        var report = base.CreateDispatchReportAsPdf(dispatch);
        var stationeryFile = appDataFolder.RetrieveBytes(stationeryFileName);
        var result = pdfService.Stamp(report, stationeryFile);
        return result;
    }
    //...
}

```

To add appendices (e.g. the terms and conditions) the *Merge* method of the IPdfService can be used:

```

c#
public class CustomServiceOrderService : ServiceOrderService, IServiceOrderService, IReplaceRegistration<ServiceOrderService>
{
    public override byte[] CreateDispatchReportAsPdf(ServiceOrderDispatch dispatch)
    {
        var report = base.CreateDispatchReportAsPdf(dispatch);
        var termsAndConditionsFile = appDataFolder.RetrieveBytes(termsAndConditionsFileName);
        var result = pdfService.MergeFiles(report, termsAndConditionsFile);
        return result;
    }
    //...
}

```

14.6 Changing the view

The dispatch report consists of three views which can be overridden individually to match the customer needs:

- *src\Crms.Web\Plugins\Crms.Service\Views\Shared\DispatchReport.ascx* - the main part of the report. This view is also used as the report preview on the mobile service client
- *src\Crms.Web\Plugins\Crms.Service\Views\Dispatch\ReportHeader.aspx* - the header of the report which will be repeated on every page
- *src\Crms.Web\Plugins\Crms.Service\Views\Dispatch\ReportFooter.aspx* - the footer of the report which will also be repeated on every page

15 Optimize performance for web applications

All recommendations presented here are based on experience for a complex application based on Asp.Net and the associated data structure (e.g. Crm or SMS, possibly with offline components).

15.1 Database

- SQL Server Express limitations through full SQL Server. SQL Server Express can only allocate 1GB memory per instance ([Microsoft comparasion](#))
- Memory, memory, storage and a bit of CPU ie at least 8GB, 4 cores or better.

"If your server is slow and you've got less than 64GB of memory, learn how to explain business costs as I explain in the video. It's smarter to buy \$500 worth of memory rather than spend days struggling with a problem and making risky changes to your server. Sure, the business is going to argue against you – that's their job, and it's your job to clearly explain the pros and cons of each side" via [Brentozar](#)

- Physically separate the database server and web server but connect them with high performance (e.g. Gigabit Ethernet)
- Look for suspicious queries, that are, profiling the database and look for any suspicious pattern. For example with the help of [SQL Express Profiler](#)
- The right indices are especially important for complex queries, for this there are a few sources:
 - <http://www.brentozar.com/sql/index-all-about-sql-server-indexes/>

```
-- Create demo table
CREATE TABLE Sales(
    ID INT IDENTITY(1,1)
    ,ProductCode VARCHAR(20)
    ,Price FLOAT(53)
    ,DateTransaction DATETIME);

-- Create test data proc
CREATE PROCEDURE InsertIntoSales
AS
SET NOCOUNT ON
BEGIN
DECLARE @PC VARCHAR(20)='A12CB'
DECLARE @Price INT = 50
DECLARE @COUNT INT = 0
    WHILE @COUNT<200000
    BEGIN
        SET @PC=@PC+CAST(@COUNT AS VARCHAR(20))
        SET @Price=@Price+@COUNT
        INSERT INTO Sales VALUES (@PC,@Price,GETDATE())
        SET @PC='A12CB'
        SET @Price=50
        SET @COUNT=@COUNT+1
    END
END

-- execute it
EXEC InsertIntoSales

SET STATISTICS IO ON

-- Read without index, watch logical reads
SELECT * FROM Sales WHERE ID=189923
-- Create clustered Index on ID Column
CREATE CLUSTERED INDEX CL_ID ON SALES(ID);
-- Now Re-Read with index, watch logical reads drop to 3
SELECT * FROM Sales WHERE ID=189923

--- Non-Clustered Index ---

-- Read without index, watch logical reads
SELECT * FROM Sales WHERE ProductCode like 'A12CB908%' ORDER BY Price
-- Create nonclustered Index on ProductCode Column
CREATE NONCLUSTERED INDEX NONCI_PC ON SALES(ProductCode);
```

```
-- Now Re-Read with index, watch logical reads drop
SELECT * FROM Sales WHERE ProductCode like 'A12CB908%' ORDER BY Price

DROP INDEX Sales.CL_ID;
DROP INDEX Sales.NONCI_PC;
```

via [codeproject](#)

- If deadlocks occur: First try to reduce the transaction time by optimizing the queries
 - Less complex queries
 - Smaller amounts of data (e.g. through paging with e.g. TOP 25)
 - Improved indexes ie less time in the transaction
 - Check indices for insert and update as well
- Snapshot Isolation can be used to drastically reduce deadlocks.
 - Transaction-related snapshots are managed in the TempDB
 - SELECTS access a consistent database status before the last transaction

```
ALTER DATABASE MyDatabase
SET ALLOW_SNAPSHOT_ISOLATION ON

ALTER DATABASE MyDatabase
SET READ_COMMITTED_SNAPSHOT ON
```

- Maintenance of the database
 - Reduce volume periodically, for example by archiving
 - In the Crm, for example, delete the Crm.Log table on a rolling basis every 7 days

```
USE <DATABASE>

DECLARE @db AS NVARCHAR(100)
SELECT @db = DB_NAME()

IF NOT EXISTS(SELECT * FROM sys.indexes WHERE object_id = object_id('CRM.Log') AND NAME ='IX_Log_Date')
CREATE NONCLUSTERED INDEX IX_Log_Date ON [CRM].[Log] ([Date])

DELETE FROM CRM.[Log] WHERE [date] < DATEADD(day, -7, GETDATE())

DBCC SHRINKDATABASE(@db)
```

- A good option to maintain your database according to [brentozar.com](#) is to periodically call the IndexOptimize script of Ola Hallengren. See examples on the bottom of the page for more detailed explanations.

```
EXECUTE dbo.IndexOptimize @Databases = 'USER_DATABASES',
@FragmentationLow = NULL,
@FragmentationMedium = 'INDEX_REORGANIZE,INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE',
@FragmentationHigh = 'INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE',
@FragmentationLevel1 = 5,
@FragmentationLevel2 = 30,
@UpdateStatistics = 'ALL',
@OnlyModifiedStatistics = 'Y'
```

15.2 Database Maintenance

A new background agent called DatabaseCleanupAgent is implemented to our system. It runs once a night by default and deleted deprecated data from tables. The deprecation measured in days can be adjusted in the appSettings configuration of the main plugin.

```
- Maintenance/PostingDeprecationDays (int)
- Maintenance/MessageDeprecationDays (int)
- Maintenance/LogDeprecationDays (int)
- Maintenance/ErrorLogDeprecationDays (int)
- Maintenance/ReplicatedClientDeprecationDays (int)
```

Additionally the defragmentation/reorganization of indices is performed by the background agent. For this, you only have to add the prodcedures of OlaHallengren's script to your databases. The scripts are added in "tools\OlaHallengren" of your source

folder. The FragmentationLevels for the OlaHallengreen script is adjustable in the appSettings configuration.

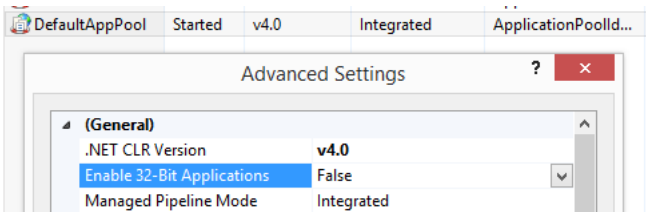
```
- Maintenance/FragmentationLevel1 (int) percentage
- Maintenance/FragmentationLevel2 (int) percentage

- Fragmentation < Level 1: Do Nothing
- Fragmentation > Level 1 && < Level 2 : Reorganize index, if not possible rebuild the index online.
- Fragmentation > Level 2: Rebuild the index online, if not possible rebuild the index offline
```

For detailed information about the configuration keys, take a look into configuration section of our dev documentation.

15.3 IIS

- Memory, memory, memory and a bit of CPU (at least 8 GB, 4 cores better more)
- 64-bit worker processes to allocate enough memory.
- 32-bit worker processes have private memory limits of approx. 2 GB (depending on the architecture), followed by OutOfMemoryExceptions (can be remedied through timely recycling).



32Bit Deactivate process

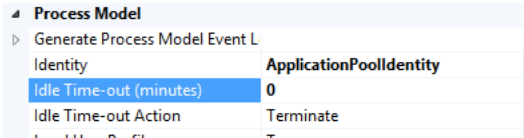
- Pre-compile views
 - ASCX and ASPX files are compiled the first time they are called by the worker process, ie access to the hard disk + memory and CPU to compile the views, runtime errors may only be noticed at this moment due to faulty views. With precompiled views, a binary version is already stored on the server. Advantage: The first access to the page is 2-3 seconds faster, depending on the memory configuration.

	activebravosforcontact.ascx.7854784f.compiled	24.07.2014 10:57	COMPILED File
	activedispatches.ascx.869f2092.compiled	24.07.2014 10:56	COMPILED File
<input checked="" type="checkbox"/>	adddocumentlink.ascx.639c3968.compiled	24.07.2014 10:55	COMPILED File
	addgeodatepopup.ascx.5279444.compiled	24.07.2014 10:56	COMPILED File
	addressaddpopup.ascx.5279444.compiled	24.07.2014 10:56	COMPILED File
	addresscommunicationseditor.ascx.83d7912c.compiled	24.07.2014 10:57	COMPILED File
	addresseditorbody.ascx.c508ba2d.compiled	24.07.2014 10:57	COMPILED File

Precompiled views

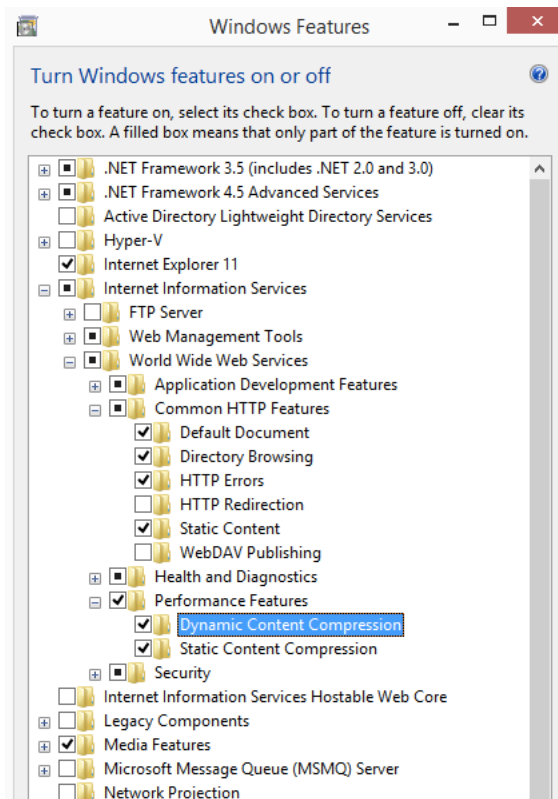
- Shut down the log level in production, info or warning and only reactivate it when necessary

- Set recycling (memory and time)
 - With 32 bit or depending on the memory configuration to avoid OutOfMemory exceptions. Nightly recycling is good.
 - Issue Recycle application pool after inactivity (otherwise the application starts within the day after 20 minutes of inactivity)



Deactivate idle timeout

- Use HTTPPing to preheat the page
 - After the restart in the night a first access to load the required resources into the memory = first access in the morning :)
- Compress and cache resources (dynamic content) = is deactivated by default in IIS and prevents, for example, remaining calls to an API from being compressed (gzip)



Compress dynamic content

15.4 Application

- Set Quartz.Net CRON Trigger correctly
 - Check Jobs.xmls in all plugins
 - Do not trigger unnecessarily often
 - Remove unnecessary triggers e.g. Dropbox or SmtptDropbox
- Use paging (e.g. for address coding) to process only limited amounts of data
- Activate profiling (? Profile = true) in the url to check the loading times of the page

15.5 programming

- Careless use of resources in the NHibernate
 - Reloading lazy objects at runtime is better where possible, eg loading collections or parent relationships with the help of eager fetch strategies
 - Frequent and thoughtless database queries
- In general, every database query is valuable, ie. as few DB round trips as possible

Example of eager fetch strategy

```
public override IQueryable<ServiceOrderHead> Eager(IQueryable<ServiceOrderHead> entities)
{
    entities.Fetch(x => x.CustomerContact);
    entities.Fetch(x => x.Payer);
    entities.Fetch(x => x.Initiator);
    entities.Fetch(x => x.ServiceObject);
    entities.Fetch(x => x.InvoiceRecipient);
    ...
}
```

16 Documentation with the Nudoc tool

Nudoc is a commandline application that wraps a [Markdown Extra](#) parser, a Razor template [engine](#) as well as a [Html to pdf conversion library](#) to produce nicely formatted Pdf files.

I use it to write technical documents. The default output is Html. Optionally it generates a Pdf document from the Print stylesheeted version of your Html.

The provided default templates are just a starting point. These may be personalized for custom, e.g. client specific documents and the like.

16.1 Directory Layout

Nudoc uses the following directory layout for your document.

```
in                --> contains the document's chapters
  chapter1
    img           --> optional image files for this chapter
    chapter1.md   --> markdown content for this chapter
  chapter2
    chapter2.md

style             --> optional css styles
js               --> optional js scripts
template         --> optional razor templates for conversion

out              --> your doc ends up here

toc.txt          --> reference of all chapters and their order
nudoc.config     --> configuration for the build process
```

16.2 Usage

Initializing a new directory for nudoc, this creates a sample document and config files

```
C:\>nudoc --init --init-dir "c:\documentation"
Initializing to c:\documentation...
```

Create document, add --pdf for Pdf output in addition to Html

```
C:\>nudoc --config "c:\documentation\nudoc.config" --build
Check environment...
Cleaning...
Removing c:\documentation\out...
Parse toc ...
Copy dependencies...
Processing input files...
Generating document...
```

For quick preview Nudoc contains an integrated web server module that allows you to preview your documentation file from the browser and rebuild it every time you reload the page

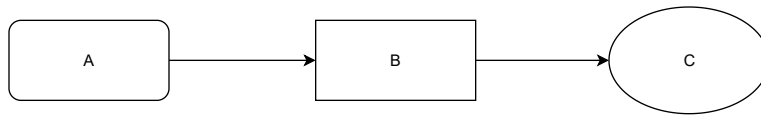
```
C:\>nudoc --config "c:\documentation\nudoc.config" --service
Check environment...
Listening to http://localhost:8080/
Press any key to stop...
```

If you want to customize the default Html template, Js files or Css style you can edit the files in `./template`, `./style` or `./js` to suit them to your needs.

If Nudoc throws an error most likely your `--config` parameter is wrong or the `nudoc.config` file format is not correct.

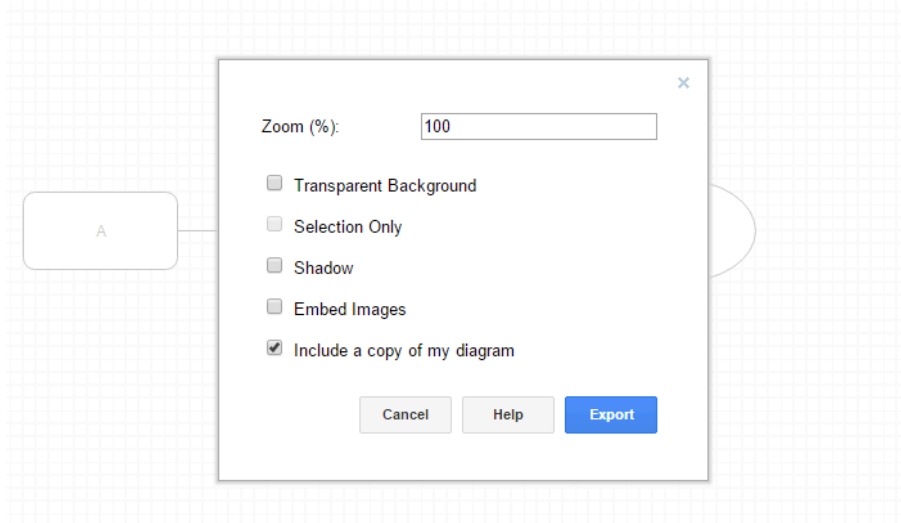
16.3 Flowcharts

Charts should be created using draw.io(<https://draw.io>) and saved as SVG (Scalable Vector Graphics). This way they can be easily resized and printed in high quality. See this SVG in the image folder of this file to see an example.



A SVG drawing created with draw.io

Please save the original SVG or Xml File alongside the output. This way anybody else can edit your diagram if needed.



Export and embed

16.4 Tips

- Nudoc scans your files for %Title: %Author: and %Version: lines. Authors will be combined into an array, the others two are exclusive (last found will be used)
- You can place a # in front of a TOC entry to disable a chapter
- Markdown language highlighting is available in my dropbox, copy this file to %APPDATA%\Notepad++ and restart your notepad++ instance.

16.5 Requirements

- Nudoc comes IL merged with all dependencies, so just xcopy it to a convenient location and you should be fine
- .Net Framework 4.5
- If you want to start Nudoc from everywhere put the folder where it resides into your PATH variable

16.6 Where to find it

- Download it from my dropbox [shared folder](#)
- Get it as part of Crm/Service default build artifacts from ci.l-mobile.com
- Build it from code using Crm.Documentation.CommandLine project

```
hg clone hg clone https://svn.l-mobile.com/Crm#default
```

16.7 Stylesheet

16.7.1 Structure

Make sure your documentation has a meaningful and simple structure that the reader may follow. In general there are few rules of thumb to make sure your documentation will simplify the usage of L-mobile products for the end user:

- **From simple to complex:** Structure your texts from simple to complex content.
- **Logical order:** Structure your steps, paragraphs and the overall document in a meaningful way.
- **Helpful headings:** Use headings with meaningful content to help structure the document.
- **Targeted references:** Use references wherever necessary, but make sure they are targeted to the most precise location.

16.7.2 Language and tone

- **Not So Imperative:** As a rule of thumb, avoid the word "must" and unnecessarily imperative statements in general. E.g., instead of "Commands must be run from a cmd or powershell prompt," try "You can run commands either from a cmd or powershell prompt."
- **Keep things simple:** Use short and useful sentences.
- **Avoid passive voice:** Past-tense passive voice is to be avoided. E.g., "the pause event fires when..." rather than "the pause event is fired when..." Rule of thumb: banish verbs ending in "ed". Passive voice is more appropriate for reference doc, you should still avoid it when there's a better active-voice alternative:
- Avoid em dash clauses--they unnecessarily break up the text--and parenthical statements (which can also be distracting). If you do include either, remove spaces around the dashes (punctuate fragments outside of paren's). (Punctuate full sentences inside of parens.)

16.7.3 Specification

Before you start to produce content for the technical documentation, make sure you understand your audience. Some questions to get you guided:

- Which advantages does the user have from using the product?
- How will the product be used?
- Which knowledge does a potential user have?
- Which knowledge does a potential user need?

In a perfect world you will deliver the required informations to your user as short, reliable and simple as possible. Please read your documentation after producing it and if possible let a potential user follow your documentation. Improve the parts where the user did not succeed or got stuck in the process.

16.7.4 Structure and Flow

- **Valid Heading Structure:** One title only (an A-head) at the top of the page. No jumps to C-heads. Do not follow a heading with a subheading with no intervening introductory text.
- **Flatten Content:** Use B-heads to divide each page's content. Please avoid C-heads. Readers tend to lose context once content gets that nested. Think: "blog post."
- **Consistent Headings:** Try to match verb tense in headings. That is, if one is task-based and reads "Adding a Platform," the other should be "Developing for Android" rather than "Android Development"
- **Listings:** Use numbered lists for sequential procedures or ranked content only, otherwise use bullet lists. Precede lists with text that introduces them, preferably ending with a colon (:). Do not allow single-item lists. Use bullet lists only to indicate a clear set of choices that you first introduce, not as a substitute for a series of regular paragraphs. Avoid nested lists, for the same reason you avoid C-heads. Do not indent top-level lists.
- **Punctuating and Formatting Lists:** Append periods to bullet and number list items that consist of full sentences. No periods when bullet items are sentence fragments, except in bullet lists that need internal consistency. To improve readability, add vertical space around top-level bullet/numbered list items that consist of full sentences. OK to vertically collapse lists of short, easily scannable items.
- **Topic/Comment Lists:** For bullet lists that provide short topics followed by comments, bold the topic, then merge the comment on the same line, separated with a colon unless the topic features trailing punctuation. This list item provides an example. Bolded Topic Text Follows Same Init-Cap Rule as Headings.
- **Highlight user interface elements::** Use *italic* for on-screen items, such as the names of menu items, buttons, and check boxes.
- **Notes:** Bold and all-cap NOTE: with horizontal lines outside, always at the end of a paragraph, never mid-paragraph. Do not incorporate this font change into a sentence as in "Note that..." but OK to start a sentence: "Note that..." when the information is less consequential.

NOTE The following sentence clarifies some of the aspects of the paragraph above.

- **Warnings/Tips:** As an alternative to NOTE:, use WARNING: for serious matters, or TIP: to pass along useful tricks or context.
- **Minimize Notes:** Avoid too many notes and warnings, as it implies a haphazard development environment with too many variable factors that divert the reader's attention. Try to clarify variations on a procedure within the text. Try to focus attention on information specific to the software, not on background knowledge.

16.7.5 Screenshots

- Screenshots should be created with (Awesome Screenshot)[http://awesomescreenshot.com]
- Save your screenshots in png format, don't compress the files
- Please don't alter the original screenshot (e.g. add markers or borders), this makes the process of creating multi language documentations much easier
- Please use english names for your Screenshots, e.g. menu_dispatch_list.png

To highlight a certain area on the screen please use visual context, crop the image so that the most important information is visible with a little bit of surrounding UI. That way you can easily direct the focus of your reader to the necessary area. If the area is part of a bigger screen, try to create a bigger picture first and reuse individual parts of it by cropping and explaining them separately.

16.7.6 Table of content

- Use a static destination address tag which is not language depending. E.g. Use as Header: ## Rollen {#user-manager.rolls} instead of plain ## Rollen. See as well [Links](#).

16.7.7 Links

- Use titles in link text, never 'click here'. Consider using the link's domain name as a substitute.
- Domain names are OK as a link's display text, but not full scheme-qualified URLs. No bare URLs in text outside of code.
- Link to headers by using static destination address tags as link text. Example for link text: #stylesheet.table-of-content
Scheme of link text: #document_name.header-link-text

16.7.8 German - Wiederkehrende Begriffe für einfachere Wiedererkennung

- Listenansicht (für GenericLists)
- Formularansicht (für das Bearbeiten und Anlegen)
- Kontextaktionen (für Primary und Secondary Actions)
- Kontextinformationen (für den Sidebarbereich)
- Navigationsbereich (für Hauptnavigation)
- Anwendungsbereich (für die primären Inhalte der Seite)

16.7.9 German - Anti-Patterns

- Anwendung statt Software, Produkt, etc.
- Formular statt Dialog
- Ausführen statt Klicken
- Schaltfläche statt Button
- Modul statt Plugin
- Lesezeichen statt Bookmark
- Geladen statt Download
- Verweis oder Referenz statt Link

17 Deployment

Remember to build the correct configuration (Debug for Test, Release for Production)

Important Validate your *.config files for correct connection strings after you grab them from the Build server artifacts.

17.1 Initial deployment

Initial deployment always will be executed for the **Test** environment.

- Create a working branch cb-x.x-customer from current stable
- Add new config transformations for customer environment in config folder
 - jobs.debug.xml + jobs.release.xml
 - web.debug.config + web.release.config
 - plugins.dat
 - any jobs..xml + appSettings..xml for active plugins
- Push customer branch
- Run customer branch build on ci.l-mobile.com and download precompiled artifacts to target Webserver
- Download Staging_Template database backup from [current stable artifacts](#)
- Copy contents of previously downloaded artifacts to LmobileTest folder

17.2 Moving Test to Production

- Nominate a responsible for the deployment if multiple developer work in a project
 - Make sure anybody knows about the deployment and is available if any problems
- Close open developments in the Test system, commit and push your changes,
- Merge open cfb-* branches if any
- Make a Backup of current LmobileProd database (if already available) and restore it to the L-mobileTest database
 - If your database is linked to any Legacy system (ERP, DMS, etc.) and any of the connection properties are stored inside your database (e.g. Stored Procedures, Views, Synonyms, etc.) you should change the settings after restoring the database and starting your application
 - Remember to change settings stored in Crm.Site table
 - Please take care if ERP System was not restored as well that order numbers could be out of sync
 - If any offline databases exist, make sure to reinstall the app to prevent data failure
- Execute migrations in the LmobileTest system and make sure they succeed
- Start LmobileTest application and compare the current status against Production (if available)
 - If data was migrated: Check UI record counts in Prod vs. Test (e.g. Dashboard, Contacts, etc.)
 - Execute smoke tests (e.g. Login, Login Mobile, Create order, dispatch order, export to Legacy System) internally or with your project manager for the most critical and most used functions
 - Consider creating a TOP 5 used function list for the customer to check on every deployment
 - Now it is time for some performance checks, does the system perform with production data, try if you can improve the system by finding bottlenecks
- Showcase new features to your customer / project manager
- **IMPORTANT** Get Approval document from customer for latest changes at least via E-Mail stating minimum Date and Name of User + written statement **Features (name features) have been tested thoroughly and are ready for Production** or similar. If there are any open issues append them to the written approval with a delivery date for fixing
- If any problems occurs during the tests, consider moving back to development phase and repeat QA
- Negotiate production deployment date
 - Ask the customer to inform all users that the Production update will be due on this date
 - Advise your customer to train the users before making a deployment
 - **Important** If system contains Offline components, please make sure users sync pending changes before the Update, so you can expect a clean state during upgrade. If the update fails the chance is you loose no pending data.

Expert Advice Don't deploy the system on your last working day before a holiday if you don't have written approval and are aware of the consequences.

The best day for deployments is a working day in the morning e.g. Tuesday morning at around 8 AM. That way you are in the office if something goes wrong. Deploying on Friday afternoon is considered bad practice, cause you and your customers are

going to the weekend but your customer is likely going to check the system on Saturday afternoon. Every bug will be a bad start on Monday mornings.

Don't deploy on Friday afternoons!

If you manage to deploy to a small group first (e.g. Project manager, customer key users, etc.), always do so, you will get feedback before things go out of control.

-
- After you have written approval and permission for deployment of your customer:
 - Create a clean release build from ci.l-mobile.com for all components.
 - Force the application offline during the planned time period by stopping the application pool
 - Make a back up of the SQL Server production database
 - Make a back up of the inetpub\wwwroot\lmobileProd folder
 - Run migrations in Production system and make sure they succeed
 - Update the inetpub folder for Production system using your clean release build artifacts
 - Start the application pool and restart the system
 - Execute smoke tests in production (e.g. Login, schedule a dispatch, create an order) for the most critical and most used functions
 - If a serious problem occurs you may consider downgrading and fixing the problems in the Test environment instead.
 - If smoke tests succeed: Write a Release note and publish it to the project participants
 - **Important** Merge your cb-x.x-customer branch to your cb-stable-x.x-customer to have a root for hotfixing. If it doesn't exist, create a new named branch cb-stable-x.x-customer
 - Grab a coffee and look at the beauty of your new features in production

17.3 Hotfixes in Production

Sometimes (hopefully not so often) customers will find bugs after you moved into production. To solve these situations easily it is important to create a cb-stable-x.x-customer after every successful deployment to the production system.

Important This branch is meant to be used for hot fixes only. Don't ever try to start building new features in here.

- Update your working copy to cb-stable-x.x-customer head revision
- Find and fix the bug using a copy of the production database
- Commit and deploy
- Validate your fix in Production and inform your customer
- Merge hotfix from cb-stable-x.x-customer to your cb-x.x-customer branch to make sure it is removed for the next iteration as well

17.4 Upgrading from release x.x to x.y

- Close all open developments in cb-x.x-customer and move them to production as described above
- Create a new branch cb-x.y-customer from current release x.y stable
- Merge latest cb-stable-x.x-customer to cb-x.y-customer resolving conflicts where necessary
- Deploy to the target Webserver Test environment using the above described steps
- After creating the new branches close previous cb-x.x-customer and cb-stable-x.x-customer branches to prevent confusion

18 Documentation of application settings

18.1 Something you need to know for this document

18.1.1 How do I identify a configuration switch as price list position?

All configurations which are price list positions can be identified by following pattern:

CRM/sales: CRM/sales: SW<6 digit number>

Service: Service: SW<6 digit number>

19 Documentation of plugin settings

19.1 Crm.Article

Enables article feature.

Please note: This plugin is required for other plugin(s). If it is required it will be added automatically by dependency resolution of the other plugin.

19.2 Crm.Campaigns (CRM/sales: SW000344)

Enables campaign feature.

19.2.1 FinishedCampaignsSelectableForDays (int)

Define if campaign is selectable for contacts after finished. Default value is 60.

19.2.2 CampaignPriceStep (int)

Define quantity step when using up and down arrows in UI. Default value is 100.

19.3 Crm.Configurator (CRM/sales: SW000347)

Please note:

When Crm.Configurator plugin will be enabled, it changes necessary lookup configuration in [LU].[OrderEntryType] automatically.

19.4 Crm.Documentation

Enables documentation accessibility.

19.5 Crm.Documentation.CommandLine

Internal usage.

19.6 Crm.DynamicForms

Enables dynamic form features. It is required e.g. for SMS.Checklists.

Please note: This plugin is required for other plugin(s). If it is required it will be added automatically by dependency resolution of the other plugin.

19.7 Crm.ErpExtension (CRM/sales: SW000531 + SW000532)

Enables ERP documents and turnover statistics. The single features can be activated separately by permissions.

Please note:

Feature	Permission Group / Permission	Price list position
ERP documents tab in company details view (sales client)	Company / ErpDocumentsTab	SW000531
ERP documents tab in project details view (sales client)	Project / ErpDocumentsTab	SW000531
Statistics tab in company details view (sales client)	Company / TurnoverTab	SW000532

19.8 Crm.InforExtension (CRM/sales: SW000267; service: SW000270)

Standard interface to ERP Infor COM.

Please note:

- A service to handle the data on Infor site needs to be installed.
- No reading interface from Infor COM to L-mobile is implemented.
- The following data can be written from L-mobile to Infor COM:
 - Companies
 - Persons
 - Communication data

19.9 Crm.Offline (CRM/sales: SW000327)

Enables offline sales client and technician client.

19.10 Crm.Order (CRM/sales: SW000345, SW000346)

Enables orders and offers.

Attention:

- SW000345 can be enabled/disabled by lookup [LU].[OrderEntryType]. Please change IsActive of SingleDelivery appropriately.
- SW000346 can be enabled/disabled by lookup [LU].[OrderEntryType]. Please change IsActive of MultiDelivery appropriately.

Please note: Crm.Configurator changes its lookup in [LU].[OrderEntryType] automatically when activating the configurator plugin.

19.11 Crm.Project (CRM/sales: SW000342)

Enables project module.

19.12 Crm.ProjectOrders (CRM/sales: SW000342)

Enables order integration to projects. When this plugin is enabled the corresponding orders which are created from a project will be referenced in the project. A order tab in projects will be visible to access the corresponding orders. It is also possible to create orders in projects.

19.13 Crm.Service (Service: SW000351)

Activates basic service features.

Attention:

- ServiceContract can be enabled/disabled by permission (SW000371)

PGroup: WebAPI; Permission: ServiceContract

- ServiceObject can be enabled/disabled by permission (SW000372)

PGroup: WebAPI; Permission: ServiceObject

19.14 Crm.VisitReport (CRM/sales: SW000340; SW000341)

Attention: By activating this plugin VisitReport (SW000340) and Tour planing (SW000341) functionality will be added to the application. Up from V4.5 it will be possible to activate them separately by permission.

19.15 Main (CRM/sales: SW000325; Service: SW000351)

This is the base module and required for CRM/sales and service application.

19.16 Main.SmtpDropbox (CRM/sales: SW000325; Service: SW000351)

Enables dropbox module with smtp.

19.17 Sms.Checklists (service: SW000373)

Enables dynamic form integration for checklists.

19.18 Sms.TimeManagement (service: SW000367)

Enables time management integration.

20 Documentation of configuration settings

All configuration settings of our web-configs will be grouped by plugins. Additionally only settings that are / were used by our code will be documented. Quotations aren't qualified or may be wrong descriptions and have to be checked.

20.1 Something you need to know for this chapter

20.1.1 How to create "string arrays"

Create string arrays comma separated without space.

20.2 Main

20.2.1 Address/DisplayOnlyRegionKey (bool)

If this is set to true, only in the csv export the region key is displayed instead of the region name

20.2.2 AllowCompanyTypeSelection (bool)

When set to false, the CompanyType dropdown is disabled. For a new company the company type is set to the CompanyType which is marked as favorite in the lookup table. Feature can be useful if contacts are imported from an ERP whereas the companies created in the L-mobile system should all belong to a specific company type, e.g. "Prospect".

20.2.3 ApplePushNotification/CertificateFileName (string)

Sets the certificate filename to register the apple devices to the apple push notification service.

20.2.4 ApplePushNotification/CertificatePassword (string)

Sets the password to register the apple devices to the apple push notification service.

20.2.5 ApplePushNotification/ProductionEnvironment (bool)

Defines if the certificate is used as production certificate.

20.2.6 CefToPdfPath (string)

Optional path to manually deployed CefToPdf to use with StaticDeployment. Can be used to share a CefToPdf deployment between multiple applications.

20.2.7 Configuration/BravoActiveForCompanies (bool)

Decides if bravos are available for companies.

20.2.8 Configuration/BravoActiveForPersons (bool)

Decides if bravos are available for persons.

20.2.9 CompanyGroupFlags/AreSearchable (bool)

This option adds the company group flags completely. I.E. company group flags are editable in company and filterable using contact filter.

20.2.10 CompanyNoIsGenerated(bool)

Decides if the CompanyNo is automatically generated by L-mobile

20.2.11 CompanyNoIsCreateable(bool)

Decides if a user can create a custom CompanyNo when creating a company

20.2.12 CompanyNoIsEditable(bool)

Decides if a user can edit an existing CompanyNo

20.2.13 Cordova/AndroidAppLink (string)

Sets the link at login to I-mobile client app in google playstore.

20.2.14 Cordova/AppStoreAppLink (string)

Sets the link at login to I-mobile client app in apple appstore.

20.2.15 Cordova/Windows10AppLink (string)

Will set the link at login to I-mobile client for window appstore.

20.2.16 DropboxForwardPrefixes (string)

This defines the prefixes of the email subject to detect forwarded emails.

20.2.17 DropboxDomain (string)

This key sets the domain of the user dropbox address as well as of the project dropboxaddress. E.g: key="crm.example.com" results as (6e732bc93ad@crm.example.com)

20.2.18 MinFileSizeInBytes (int)

This key is used to limit the storage of attached file belonging to Dropbox Email Note

20.2.19 MinPasswordStrength (int)

Minimum password strength score from 0 to 4 (inclusive) when a user changes the password, with 0 being least secure and 4 being most secure, calculated from the number of guesses estimated to be needed.

20.2.20 DropboxLogMessages (bool)

Decides if the recieved dropbox messages should be logged to disk (always, not just in case of error).

20.2.21 FileResource/AllowedContentTypes (string[])

Optional whitelist of allowed content types for file uploads. If empty all content types are allowed.

20.2.22 FileResource/ContentTypesOpenedWithoutSandbox (string[])

Optional whitelist of allowed content types which will be opened directly instead of rendered inside a sandboxed iframe. If empty all content types are allowed. All image types can be allowed by using image/* like in html inputs

20.2.23 Geocoder/GeocoderService (string)

Defines which geocoding service is used.

Options:

- "mapquest"
- "bing"
- "yahoo"
- "google"

Default value: The default is google, even if the GeocoderService string is empty.

20.2.24 Geocoder/BingMapsApiKey (string)

Contains the api key for Bing maps.

20.2.25 Geocoder/GoogleMapsApiKey (string)

Contains the api key for Google maps.

20.2.26 Geocoder/MapQuestApiKey (string)

vContains the api key for Map Quest.

20.2.27 Geocoder/YahooMapsApiKey (string)

Contains the api key for Yahoo maps.

20.2.28 Geocoder/YahooMapsApiSecret (string)

Contains the api key secret for Yahoo maps.

20.2.29 Lucene/LegacyNamelsDefault (bool)

Decides if contacts are filterable for their legacyId using quick search (e.g. filter in right upper corner of dashboard).

20.2.30 MapTileLayerUrl (string)

URL for tiles for leaflet maps.

20.2.31 PasswordReset/ExpirationInMinutes (int)

Duration in minutes a password reset email is valid

20.2.32 PasswordReset/MaxEmailsPerHour (int)

Maximum amount of password reset emails a user can send each hour

20.2.33 Person/BusinessTitlesLookup (bool)

Decides if the businesstitle of persons is a lookup or a free text to set.

20.2.34 Person/DepartmentIsLookup (bool)

Decides if the department of persons is a lookup or a free text to set.

20.2.35 PersonNolsGenerated(bool)

Decides if the PersonNo is automatically generated by L-mobile

20.2.36 PersonNolsCreateable(bool)

Decides if a user can create a custom PersonNo when creating a person

20.2.37 PersonNolsEditable(bool)

Decides if a user can edit an existing PersonNo

20.2.38 RedisConfiguration (string)

Configuration command string for redis. If empty a local standalone redis server will be started.

20.2.39 Report/HeaderHeight (double)

Sets the header height of reports in cm.

20.2.40 Report/HeaderSpacing (double)

Sets the content spacing of report headers in cm.

20.2.41 Report/FooterHeight (double)

Sets the footer height of reports in cm.

20.2.42 Report/FooterSpacing (double)

Sets the content spacing of the report footers in cm.

20.2.43 Site/HostEditable (bool)

When set to true, the site host can be edited in the site settings.

20.2.44 Site/PluginsEditable (bool)

When set to true, the active plugins can be edited in the site settings.

20.2.45 StripLeadingZerosFromLegacyId (bool)

When set to true, leading zeros from Legacy Ids are removed when being displayed.

20.2.46 UseActiveDirectoryAuthenticationService (bool; CRM/sales: SW000336; Service: SW000366)

Enables active directory authentication for login.

20.2.47 ActiveDirectoryEndpoint

Connection string to use for active directory authentication.

20.2.48 Maintenance/PasswordResetTokenDeprecationDays (int)

Password reset tokens are deleted after n days by DatabaseCleanupAgent. (default 30)

20.2.49 Maintenance/PostingDeprecationDays (int)

Postings that are 'Processed', 'Skipped' or 'Stale' are deleted after n days by DatabaseCleanupAgent. (default 90)

20.2.50 Maintenance/MessageDeprecationDays (int)

Messages that are sent (state == 1) are deleted after n days by DatabaseCleanupAgent. (default 30)

20.2.51 Maintenance/LogDeprecationDays (int)

Info- and warninglogs are deleted after n days by DatabaseCleanupAgent. (default 14)

20.2.52 Maintenance/ErrorLogDeprecationDays (int)

Errorlogs are deleted after n days by DatabaseCleanupAgent. (default 30)

20.2.53 Maintenance/ReplicatedClientDeprecationDays (int)

Replicated Clients are deleted by DatabaseCleanupAgent, if the LastSync is older than n days. (default 90)

20.2.54 Maintenance/FragmentationLevel1 (int 0-100)

If an index got a fragmentation above n % it will be reorganized or rebuild online. (default 5)

20.2.55 Maintenance/FragmentationLevel2 (int 0-100)

If the fragmentation of an index is above n % it will be rebuild online, if not possible it will be rebuild offline. (Table is not accessible during this time) (default 15)

20.2.56 Maintenance/CommandTimeout (int)

Defines how long the cleanup operations can take before returning a timeout in seconds. (default 1800)

20.3 Main

20.3.1 Posting/MaxRetries (int)

Maximum amount of retries for each Posting to be processed.

20.3.2 Posting/RetryAfter (int)

Amount of time in *minutes* between two tries.

20.4 Main.SmtpDropbox

20.4.1 SmtpDropboxAgent/Port (int)

Defines the port, SmtpDropboxAgent is listening on.

20.5 Crm.Campaigns

20.5.1 CampaignNoIsGenerated(bool)

Decides if the CampaignNo is automatically generated by L-mobile

20.5.2 CampaignNoIsCreateable(bool)

Decides if a user can create a custom CampaignNo when creating a campaign

20.5.3 CampaignNoIsEditable(bool)

Decides if a user can edit an existing CampaignNo

20.6 Crm.ErpExtension

20.6.1 EnableAddressExport (bool)

If set true, addresses are exported to erp manually or via LegacyUpdateService.

20.6.2 EnableCompanyExport (bool)

If set true, companies are exported to erp manually or via LegacyUpdateService.

20.6.3 EnableCommunicationExport (bool)

If set true, communications are exported to erp manually or via LegacyUpdateService.

20.6.4 EnablePersonExport (bool)

If set true, persons are exported to erp manually or via LegacyUpdateService.

20.6.5 ErpSystemID (string)

Is used to locate the erp system in ObjectLinks. The ErpSystemId is located in vpps.ini (according to infor).

20.6.6 ErpSystemName (string)

Additional to ErpSystemId, in SAP the ErpSystemName is used in ObjectLinks to open documents in the erp system.

20.6.7 ObjectLinkIntegration (string)

Provides type of ObjectLink, used to open documents in the erp system.

Options:

- "InforObjectLink" for infor
- "D3Link" for d.velop dms
- "SapLink" for sap

20.6.8 TurnoverCurrencyKey (string)

Sets the currency for turnover according to LU.Currency.

20.7 Crm.InforExtension

20.7.1 InforExport/InforErpComVersion (string)

Used to pick a suitable adapter for Infor version.

Options:

- "6.3"
- "7.1"

20.7.2 InforExport/ShortFieldNames (bool)

Has to be set true if oracle version is below 10. If so, fieldnames aren't allowed to exceed 16 characters.

20.8 Crm.Order

20.8.1 OffersEnabled (bool) (CRM/sales: SW000345, SW000346)

Defines if offers are available or not.

20.8.2 OrderBarcodeEnabled (bool)

Adds the possibility to add items to order via barcode scanner.

20.8.3 OrderBillingAddressEnabled (bool)

If true, a billing address can be set in orders.

20.8.4 OrderComissionEnabled (bool)

If true, a comission number can be added to orders.

20.8.5 OrderDeliveryAddressEnabled (bool)

If true, a delivery address can be set in orders.

20.8.6 OrderItemDiscountEnabled (bool)

Enables possibility to set discounts on order positions.

20.8.7 OrderPrivateDescriptionEnabled (bool)

Adds a private description to orders.

20.8.8 OrderSignatureEnabled (bool)

If true, a signature input is added to orders.

20.8.9 Order/OrderNolsGenerated(bool)

Decides if the OrderNo is automatically generated by L-mobile

20.8.10 Order/OrderNolsCreateable(bool)

Decides if a user can create a custom OrderNo when creating an order

20.8.11 Order/OrderNolsEditable(bool)

Decides if a user can edit an existing OrderNo

20.8.12 Offer/OfferNolsGenerated(bool)

Decides if the OfferNo is automatically generated by L-mobile

20.8.13 Offer/OfferNolsCreateable(bool)

Decides if a user can create a custom OfferNo when creating an offer

20.8.14 Offer/OfferNolsEditable(bool)

Decides if a user can edit an existing OfferNo

20.8.15 PDFHeaderMargin (double)

Sets the header margin of pdf in cm.

20.8.16 PDFFooterMargin (double)

Sets the footer margin of pdf in cm.

20.8.17 PDFFooterTextPush (double)

Sets the distance of the footer of PDF in cm.

20.9 Crm.Project

20.9.1 Configuration/BravoActiveForProjects (bool)

Activates bravos for projects, available in project details sidebar.

20.9.2 ProjectsHaveAddresses (bool)

If set to true, in project details the StandardAddress of the Project instead of the StandardAddress of the Parent is displayed.

20.10 Crm.PerDiem

20.10.1 Email/PerDiemReportApprovers (string)

Leaders who will receive email notification from PerDiemReportApprovalNotifier about Per Diem reports with *Request closed* status waiting for close approval. List of emails separated by comma.

20.10.2 Email/PerDiemReportRecipients (string)

Recipients who will receive closed Per Diem reports sent by PerDiemReportSenderAgent. List of emails separated by comma.

20.10.3 Email/SendPerDiemReportToResponsibleUser (bool)

Decides whether the perdiem report will be sent to the responsible user.

20.10.4 Expense/ClosedExpensesHistorySyncPeriod (int)

Selects expenses x days from past to sync them to service client.

20.10.5 Expense/MaxDaysAgo (int)

Sets the maximum of selectable days in the past at creating / editing expenses.

20.10.6 PerDiemReport/ShowClosedReportsSince (int)

Shows closed per diem reports which are at most x days old only.

20.10.7 TimeEntry/MinutesInterval (int)

Alters the interval of minutes selection for timeentries.

Default value:

5

20.10.8 TimeEntry/AllowOverlap (bool)

Enables to book timeentries with overlap.

Default value:

false

20.10.9 TimeEntry/ClosedTimeEntriesHistorySyncPeriod (int)

Selects timeentries x days from past to sync them to service client.

20.10.10 TimeEntry/DefaultStart (string)

Sets a default start time for time entries.

Format:

hh:mm

20.10.11 TimeEntry/MaxDaysAgo (int)

Sets the maximum of selectable days in the past at creating / editing timeentries.

20.10.12 TimeEntry/DefaultWorkingHoursPerDay (int)

Default value for user working hours per day. Used in all-day time entry creation. Default value is 8. WorkingHoursPerDay user config is using this value when create a new user.

20.10.13 TimeEntry/ShowTimeZone (bool)

If set to true, it allows the user to select timezone for time recording related entities.

20.11 Crm.Project

20.11.1 Potential/PotentialNolsGenerated(bool)

Decides if the PotentialNo is automatically generated by L-mobile

20.11.2 Potential/PotentialNolsCreateable(bool)

Decides if a user can create a custom PotentialNo when creating a potential

20.11.3 Potential/PotentialNolsEditable(bool)

Decides if a user can edit an existing PotentialNo

20.11.4 ProjectNolsGenerated(bool)

Decides if the ProjectNo is automatically generated by L-mobile

20.11.5 ProjectNolsCreateable(bool)

Decides if a user can create a custom ProjectNo when creating a project

20.11.6 ProjectNolsEditable(bool)

Decides if a user can edit an existing ProjectNo

20.12 Crm.Service

20.12.1 AdHoc/AdHocNumberingSequenceName (string)

Required to retrieve next formatted adhoc order no.

Default value:

"SMS.ServiceOrderHead.AdHoc"

20.12.2 Dispatch/DispatchNolsGenerated(bool)

Decides if the DispatchNo is automatically generated by L-mobile

20.12.3 Dispatch/DispatchNolsCreateable(bool)

Decides if a user can create a custom DispatchNo when creating a dispatch

20.12.4 Dispatch/DispatchNolsEditable(bool)

Decides if a user can edit an existing DispatchNo

20.12.5 Dispatch/SuppressEmptyMaterialsInReport(bool)

Decides if the materials/costs list should be hidden in the report when it is empty and no lumpsum entries are present

20.12.6 Dispatch/SuppressEmptyExpensePostingsInReport(bool)

Decides if the expense postings list should be hidden in the report when it is empty and no lumpsum entries are present

20.12.7 Dispatch/SuppressEmptyErrorTypesInReport(bool)

Decides if the error types and causes should be hidden in the report when it is empty

20.12.8 Dispatch/SuppressEmptyTimePostingsInReport(bool)

Decides if the time postings list should be hidden in the report when it is empty and no lumpsum entries are present

20.12.9 Dispatch/SuppressExpensePostingsInReport(bool)

Decides if the expense postings list should be hidden in the report

20.12.10 Dispatch/SuppressErrorTypesInReport(bool)

Decides if the errors list should be hidden in the report

20.12.11 Dispatch/SuppressEmptyJobsInReport(bool)

Decides if the job group header should be hidden in the report when both material and time posting lists are empty and no lumpsum entries are present

20.12.12 Email/ClosedByRecipientForReplenishmentReport (bool)

If true, adds the user, which closed the replenishment order to the recipient list of report.

20.12.13 Email/DispatchReportRecipients (string array)

Defines the recipients of the dispatch report. It also defines the recipients of serviceorder report. As possible value e.g: "default@example.com,german@example.com"

20.12.14 Email/ReplenishmentOrderRecipients (string array)

Add recipients to replenishmentorders by default. As possible value e.g: "default@example.com,german@example.com"

20.12.15 Email/SendDispatchNotificationEmails (bool)

If true, sends out emails to the dispatch user if dispatch is created with status released, or if edited from scheduled to released.

20.12.16 Email/SendDispatchRejectNotificationEmails (bool)

If true, sends out email to the dispatch responsible user if dispatch got rejected.

20.12.17 Email/SendDispatchFollowUpOrderNotificationEmails (bool)

If true, sends out email to the dispatch responsible user (or if not set, the dispatcher) if dispatch is marked with *follow up service order*.

20.12.18 Email/SendDispatchReportsOnCompletion (bool)

If true, sends out dispatch report to all dispatch report recipients by DispatchReportSenderAgent.

20.12.19 Email/SendDispatchReportToDispatcher (bool)

If true, adds dispatch crateuser to internal recipient list.

20.12.20 Email/SendDispatchReportToTechnician (bool)

If true, adds dispatched user to internal recipient list.

20.12.21 Email/SendServiceOrderReportsOnCompletion (bool)

If true, sends out serviceorder report by ServiceOrderReportSenderAgent.

20.12.22 Email/SendServiceOrderReportToDispatchers (bool)

If true, sends out serviceorder report to all createusers of attached dispatches.

20.12.23 Export/ExportDispatchReportsControlFileExtension (string)

Defines the file extension of control file.

20.12.24 Export/ExportDispatchReportsControlFileContent (string)

Defines the pattern for control file content.

Default value:

""Controlfile""

20.12.25 Export/ExportDispatchReportsControlFilePattern (string)

Defines the filename pattern for the report control file.

Default value:

"{Date-yyyy}{Date-MM}{Date-dd}_{DispatchId}"

20.12.26 Export/ExportDispatchReportsFilePattern (string)

Defines filename pattern for reports saved to disk.

Default value:

"{Date-yyyy}{Date-MM}{Date-dd}_{DispatchId}"

20.12.27 Export/ExportDispatchReportsOnCompletion (bool)

If true, saves all dispatch reports to the configured path, where it's status is "Closed" or "ClosedNotComplete"

20.12.28 Export/ExportDispatchReportsPath (string)

Defines the path where dispatch reports are saved to disk.

Default value:

"\\unc\path\{Date-yyyy}\"

20.12.29 Export/ExportServiceOrderReportsOnCompletion (bool)

If true, saves all serviceorder reports to the configured path, where it's serviceorder status is "Closed".

20.12.30 Export/ExportServiceOrderReportsPath (string)

Defines the export path where serviceorder reports are saved to disk.

20.12.31 Export/ExportServiceOrderReportsUncDomain (string)

Domain for network access.

20.12.32 Export/ExportServiceOrderReportsUncPassword (string)

Password for network access.

20.12.33 ExportServiceOrderReportsUncUser (string)

User for network access.

20.12.34 ReplenishmentOrder/ClosedReplenishmentOrderHistorySyncPeriod (int)

Selects closed replenishment orders x days from past to sync them to service client.

20.12.35 InstallationNolsGenerated(bool)

Decides if the InstallationNo is automatically generated by L-mobile

20.12.36 InstallationNolsCreateable(bool)

Decides if a user can create a custom InstallationNo when creating an installation

20.12.37 InstallationNolsEditable(bool)

Decides if a user can edit an existing InstallationNo

20.12.38 Service/Dispatch/Requires/CustomerSignature (bool)

If true, dispatches can't be completed in service client if it's not signed by customer.

20.12.39 Service/Dispatch/Show/EmptyTimesOrMaterialsWarning (string)

"NONE", "WARN" and "ERROR" are the acceptable values. Decides if a warning message will be displayed when someone tries to save a dispatch without any material and time posting. NONE stands for no warning at all, WARN means a warning message will be displayed, but the dispatch can be saved, ERROR will block the save completely if the warning message presents.

20.12.40 ServiceCase/OnlyInstallationsOfReferencedCustomer (bool)

If true, only installations, referenced by customer are available in service cases. Selected service object will override this.

20.12.41 ServiceCase/Signature/Enable/Originator (bool)

Enables signature for originator at choosing signing person in service client.

20.12.42 ServiceCase/Signature/Enable/Technician (bool)

Enables signature for technician at choosing signing person in service client.

20.12.43 ServiceCase/ServiceCaseNolsGenerated(bool)

Decides if the ServiceCaseNo is automatically generated by L-mobile

20.12.44 ServiceCase/ServiceCaseNolsCreateable(bool)

Decides if a user can create a custom ServiceCaseNo when creating a service case

20.12.45 ServiceCase/ServiceCaseNolsEditable(bool)

Decides if a user can edit an existing ServiceCaseNo

20.12.46 ServiceContract/OnlyInstallationsOfReferencedCustomer (bool)

If true, only installations, referenced by customer are available in service contracts. Selected service object will override this.

20.12.47 ServiceContract/MaintenanceOrderGenerationMode (string)

Available options: JobPerInstallation, OrderPerInstallation.

20.12.48 ServiceContract/CreateMaintenanceOrderTimeSpanDays (int)

Default time span when evaluate and generate maintenance orders.

20.12.49 ServiceContract/MaintenancePlan/AvailableTimeUnits (string array)

Selectable time units for maintenance plan creation.

20.12.50 ServiceContract/ReactionTime/AvailableTimeUnits (string array)

Selectable time units for reaction times.

20.12.51 ServiceContract/ServiceContractNolsGenerated(bool)

Decides if the ServiceContractNo is automatically generated by L-mobile

20.12.52 ServiceContract/ServiceContractNolsCreateable(bool)

Decides if a user can create a custom ServiceContractNo when creating a service contract

20.12.53 ServiceContract/ServiceContractNolsEditable(bool)

Decides if a user can edit an existing ServiceContractNo

20.12.54 ServiceObject/ObjectNolsGenerated(bool)

Decides if the ObjectNo is automatically generated by L-mobile

20.12.55 ServiceObject/ObjectNolsCreateable(bool)

Decides if a user can create a custom ObjectNo when creating a service object

20.12.56 ServiceObject/ObjectNolsEditable(bool)

Decides if a user can edit an existing ObjectNo

20.12.57 ServiceOrderMaterial/CreateReplenishmentOrderItemsFromServiceOrderMaterial (bool)

If true, the technician is able to add / update serviceorder material item to replenishment order.

20.12.58 ServiceOrderMaterial/ShowPricesInMobileClient (bool)

If true, serviceorder material prices are shown in technician client.

20.12.59 ServiceOrder/DefaultDuration (string)

Sets a default duration to adhoc orders at creation.

Format:

hh:mm

20.12.60 ServiceOrder/OnlyInstallationsOfReferencedCustomer (bool)

If true, only installations, referenced by customer are available in service orders. Selected service object will override this.

20.12.61 ServiceOrder/GenerateAndAttachJobsToUnattachedTimePostings (bool)

If true (default), new service order times (jobs) will be created or existing ones matches to newly added time postings. If false, time postings will not be automatically attached to jobs.

20.12.62 ServiceOrder/OrderNolsGenerated(bool)

Decides if the OrderNo is automatically generated by L-mobile

20.12.63 ServiceOrder/OrderNolsCreateable(bool)

Decides if a user can create a custom OrderNo when creating a service order

20.12.64 ServiceOrder/OrderNolsEditable(bool)

Decides if a user can edit an existing OrderNo

20.12.65 ServiceOrderDispatch/ReadGeolocationOnDispatchStart (bool)

If true, geolocation is saved in dispatch when technician has started working on order.

20.12.66 ServiceOrderExpensePosting/UseArticleAsExpenseType

If true Articles are used as a ExpenseType. Usually this should be true when a ERP system is used and the service order expenses should be exported to the ERP System. When false, the value from the LU.ExpenseType table is used.

20.12.67 ServiceOrderTimePosting/ClosedTimePostingsHistorySyncPeriod (int)

Selects timepostings x days from past to sync them to service client.

20.12.68 ServiceOrderTimePosting/MaxDaysAgo (int)

Sets the maximum of selectable days in the past at creating / editing timepostings.

20.12.69 ServiceOrderTimePosting/ShowTechnicianSelection (bool)

Enables technician selection in service client on creating timepostings.

20.12.70 ServiceOrderTimePosting/MinutesInterval (int)

Alters the interval of minutes selection for serviceorder timepostings.

Default value:

5

20.12.71 ServiceOrderTimePosting/AllowOverlap (bool)

Enables to book timepostings with overlap.

Default value:

false

20.12.72 UserExtension/OnlyUnusedLocationNosSelectable (bool)

If true, in user administration, only unused locations are selectable to users.

20.12.73 AttentionTasks/CheckAllInstallationsWarranties (bool)

Specify if the system have to check all installation warranties before creating the attention tasks of the service order or not.

Default value:

false

20.13 Crm.VisitReport

20.13.1 DefaultVisitTimeSpanHours (double)

Default value for visits.

20.13.2 Visit/AvailableTimeUnits (string array)

AvailableTimeUnits (comma separated). The list of selectable time units for Company visit requery. Default values: *Year, Quarter, Month, Week*

20.14 Sms.Checklists

20.14.1 Dispatch/CustomerSignatureValidateChecklists (bool)

If *true* Checklists are validated when customer signs dispatch.

20.15 Sms.Scheduler

20.15.1 WorkingTime/FromDay (int)

Sets the first day of week for the Timeline. The value is a zero based index of day of week, 0 = Sunday.

20.15.2 WorkingTime/ToDay (int)

Sets the last day of week for the Timeline. The value is a zero based index of day of week, 0 = Sunday.

20.15.3 WorkingTime/FromHour (int)

Fallback value for start of working time displayed on the Timeline. Valid values are [0;24]

20.15.4 WorkingTime/ToHour (int)

Fallback value for end of working time displayed on the Timeline. Valid values are [0;24]

20.15.5 WorkingTime/MinutesInterval (int)

Allows to specify the magnitude of a time step used in the timepickers for working times

20.15.6 WorkingTime/IgnoreWorkingTimesInEndDateCalculation (bool)

If true the events on the Timeline will ignore working times when calculating their end dates.

20.15.7 ServiceOrderZipCodeAreaLength (int)

Indicates the number of characters to use as ZipCodeArea

20.15.8 DispatchesAfterReleaseAreEditable (bool)

If true the Planning board will allow editing Dispatches after they've been released

20.15.9 DashboardCalendar/ShowAbsencesInCalendar (bool)

If true (default) the Absences created in the Planning board will be displayed on the Dashboard's calendar

20.16 Sms.TimeManagement

20.16.1 GeolocationGetCurrentPositionTimeout (int)

Defines time, after position locating request times out. Value should be in *millisecond*.

Main ### MaxFileLengthInKb (int)

Set the maximum file size to upload. Attention: Request length limited by httpRuntime maxRequestLength="..."

20.16.2 Task/AttentionTaskTypeKey (string)

Contain the key for the Attention task type

Default value:

106

21 API

21.1 Introduction

The API is designed around the [REST](#) principles using the [oData](#) protocol. HTTP response codes and messages are used to indicate API errors. To access the API, just go to:

```
subdomain.yourdomain.com/virtualpath/api
```

This document does not aim to describe all oData specifics but is a starting point and describes the limitations and special features of our implementation.

21.2 Supported Formats

The meta data model is provided in XML. Everything else is provided in JSON.

21.3 HTTP Request Methods

- GET is used for retrieving resources
- POST is used for creating resources
- PUT is used for updating resources
- PATCH is used for updating resources with partial data, it is **not** supported
- DELETE is used for deleting resources

21.4 Tools

There are several tools for different platforms to help you during API development. Since cURL is commonly used and other programs can also import requests in cURL syntax, this document will use cURL to demonstrate requests.

- [cURL](#)
- [Postman](#)
 - also available as [Google Chrome extension](#) (now deprecated)
 - import cURL requests using the button *Import* and then *Paste Raw Text*
- [RESTClient](#)
 - for Firefox
 - import cURL requests using the button *Paste* in the section *Curl* near the bottom
- [Fiddler](#) is a powerful web debugger

21.5 Authentication

There are no special permissions or roles to access the API. It can be accessed by any valid user. Authorization is done on entity level. However it is recommended to create a user that has only the required API permissions if you want to implement a recurring job like an import or an export. To authenticate there are two supported methods: the regular user credentials and the user token.

Using cookies would also work, but there currently is no defined way to retrieve the cookie.

21.5.1 Basic Authentication

You can either set the login as clear text or use the basic authentication header (which is also clear text, Base64 encoded).

```
curl -X GET "/api/Crm_Company" -H "Authorization: Basic ZGVmYXVsdEBleGFtcGxlMnVbTpkZWZhdWx0"  
curl -X GET "/api/Crm_Company" -u "default@example.com:default"
```

21.5.2 Token Authentication

Simply add your token as query parameter token to your request.

```
curl -X GET "/api/Crm_Company?$top=1&token=9f7041aa60e44b3"
```

21.6 User Agent

Please set a meaningful User-Agent HTTP header for all requests. Currently this is not required, but it may be in the future. Setting this header helps when analysing problems.

```
curl -X GET "/api/Crm_Company?token=9f7041aa60e44b3" -H "User-Agent: Paul's Test Client for Demo L-mobile v0.1"
```

21.7 Metadata

The service document, describing the endpoints, can be accessed by URI `/api/`. There you will find the link to the metadata model, which is `/api/$metadata`. The meta data model describes the entity sets, types, relationships, default values and restrictions.

21.7.1 Metadata Annotations

Type properties reflect database and business rule restrictions where possible. Some properties are marked as `Nullable="false"` and/or have a maximum allowed length. Furthermore they may be annotated with additional information. Please note that the restrictions (e.g. nullability, maximum length) are auto generated and may not be complete.

- `Lmobile.DefaultValue` describes the default value for a property, you can use this for initialization of your entities
- `InsertRestrictions`, `UpdateRestrictions` and `DeleteRestrictions` are used to mark properties that are read only. Properties with these annotations will be ignored in write operations. Some read only properties can still be written using *Direct Save* (see further below). Please note, that some properties may still be read only, even though they are not annotated.
- `InsertRestrictions`, `UpdateRestrictions` and `DeleteRestrictions` are used to mark types as restricted regarding the operations that are allowed on them. Trying to execute a restricted operation will yield an error. Some restricted types can still be written using *Direct Save* (see further below).
- `Lmobile.Required` marks a property that is required by a business rule. As some business rules are more complex, there may be properties that are required, but are not marked with the annotation.

There are entity sets with restrictions regarding the queries that they support. These entity sets have one or many of the following restriction annotations:

- `Org.OData.Capabilities.V1.FilterRestrictions`
- `Org.OData.Capabilities.V1.SortRestrictions`
- `Org.OData.Capabilities.V1.CountRestrictions`
- `Org.OData.Capabilities.V1.NavigationRestrictions`
- `Org.OData.Capabilities.V1.ExpandRestrictions`

21.7.2 Extension Values

In the meta data model you will find complex types called extension values. These are our way of extending data models from plugins. After they are defined in a plugin they will also be available in the meta data model of the API. Additionally to their extensible nature they are open types, which means you can add extra data that is not defined in the meta data model.

21.8 API Explorer

We provide an API explorer to play around with the models and operations. You can find it at:

```
subdomain.yourdomain.com/virtualpath/api/swagger
```

21.9 Reading

Reading is always done with HTTP method GET. Entities can be read as a list or as a single entity.

For clarity the following examples only contain the URI.

21.9.1 Single Entity

Entities can be directly accessed by their key:

```
/api/Crm_Company/01234567-89AB-CDEF-0123-456789ABCDEF  
/api/Crm_Company(01234567-89AB-CDEF-0123-456789ABCDEF)
```

You can also access (navigation) properties using this syntax:

```
/api/Crm_Company(01234567-89AB-CDEF-0123-456789ABCDEF)/StandardAddress
```

21.9.2 Collection of Entities

A collection can be read by using the path of the entity set:

```
/api/Crm_Company
```

To access navigational properties, you add an \$expand clause:

```
/api/Crm_Company?$expand=StandardAddress
```

To select only certain fields, use the \$select clause:

```
/api/Crm_Company?$select=Id,Name
```

You can also mix \$expand and \$select:

```
/api/Crm_Company?$select=Id,Name,Addresses&$expand=Addresses($select=City)
```

For limiting the amount of data you can use \$top and \$skip:

```
/api/Crm_Company?$skip=0&$top=10
```

Use orderby to sort the results:

```
/api/Crm_Company?$expand=Addresses($select=CreateDate;$orderby=CreateDate)&$orderby=Name desc&$select=Name
```

21.9.3 Filtering

Filtering is done by using \$filter. There are a lot of possible operators, so you will only see a few examples here. You can read more about \$filter in the [url conventions](#). Please note, that not all options are supported.

A basic filter using contains on a string:

```
/api/Crm_Company?$filter=contains(Name, 'GmbH')
```

Filtering a collection can be done using any or all:

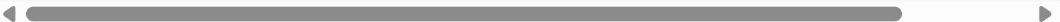
```
/api/Crm_Company?$filter=Tags/any(t: t eq 'Infor')
```

As *Tags* are a collection of strings, the same result can be achieved using in:

```
/api/Crm_Company?$filter='Infor' in Tags
```

Getting company name and street that have an address in Sulzbach:

```
/api/Crm_Company?$select=Name,Addresses&$expand=Addresses($select=Street,City)&$filter=Addresses/any(a: a/City
```



Filtering by enums can be done in two ways:

```
/api/Crm_Company?$filter=Visibility eq '2'  
/api/Crm_Company?$filter=Visibility eq 'Everybody'
```

21.9.4 Counting

To just get a plain number of results, you use /\$count:

```
/api/Crm_Company/$count?$filter=contains(Name, 'e')
```

You can also include the count into the result set, called inline count. Note that for any kind of counting, \$skip and \$top are ignored. This means, the following query returns the total number of results after applying the filter, and one entity because of

\$stop=1

```
/api/Crm_Company?$filter=contains(Name,'e')&$stop=1&$count=true
Result:
{
  "@odata.context": "/api/$metadata#Crm_Company",
  "@odata.count": 10,
  "value": [{ Id: [...] } ]
}
```

21.10 Writing

Writing is done using HTTP methods POST, PUT and DELETE. Writing operations are always done to single entities. When writing related entities you should always use batch requests (see below). Not all entities are writable and for some not all properties are writable.

21.10.1 Direct Save

The default way of writing is using internal services called SyncServices which, depending on the entity type, do additional validation, complex writing (like splitting entities or collections), only allow writing some special properties or even do not allow writing at all. You can skip the sync service and write directly to the database if you set the header `DirectSave` to true and also use a user that has the permission `WebAPI::DirectSave`.

Although validation will still take place when using this header, it should never be your default way of writing entities.

21.10.2 Special Entities

21.10.2.1 User

Writing a User is currently only possible using PUT and will only edit the **current** user (the one sending the request). Using *Direct Save* can circumvent this restriction. Creating a new user is currently not supported by the API.

21.10.3 Creating / Updating

Entities should always be sent as a whole, including the Id property. If properties are left out they will be set to their default value and sometimes the request will fail because the entity cannot be deserialized. Example: the value null is never valid for an Id.

Write operations are checked by business rules for validity. These rules for example check required fields and also field lengths and should give a clear error message about what's wrong.

For creating, either set the Id to a new unique value or a proper default value if you want to let the server decide.

```
curl -X POST /api/Crm_Address -H "Content-Type: application/json" --data-binary "@data.txt"
data.txt:
{
  "Id": "00000000-0000-0000-0000-000000000000",
  "Name1": "L-mobile",
  "Name2": "solutions",
  "Name3": "GmbH & Co. KG",
  "Street": "Im Horben 7",
  "ZipCode": "71560",
  "City": "Sulzbach an der Murr",
  "Country": "Germany",
  "CountryKey": "100",
  "RegionKey": "100",
  "POBox": null,
  "ZipCodePOBox": null,
  "AddressTypeKey": "1",
  "CompanyId": "01234567-89AB-CDEF-0123-456789ABCDEF",
  "Latitude": null,
  "Longitude": null,
  "LegacyId": null,
  "IsCompanyStandardAddress": false,
  "CreateDate": "0001-01-01T00:00:00Z",
  "ModifyDate": "0001-01-01T00:00:00Z",
  "CreateUser": null,
  "ModifyUser": null,
  "IsActive": true,
  "ExtensionValues": {
```

```
    "Distance": 1
  }
}
```

The server will respond with the created entity.

For updates, change to PUT and reference the entity you want to update:
`/api/Crm_Address(01234567-89AB-CDEF-0123-456789ABCDEF)`

21.10.3.1 Internal Properties

The following properties will always be ignored and are overwritten by the server:

- CreateDate (only for POST)
- ModifyDate
- CreateUser (only for POST)
- ModifyUser
- IsActive is used for soft deletion. You can disable soft deletion (`SoftDelete=false`) which makes this property unused.

You can set the value of these properties to something meaningful if you use them for sorting or the likes, but be prepared that they might have changed the next time you are requesting data.

21.10.3.2 Return Representation

It is recommended to set an additional header in case you do not need the returned entity: Prefer with value `return=minimal`. The server will then return an empty body instead of the created or updated entity. For POST it will return the following header: Location with an URL pointing to the entity.

21.10.4 Deleting

For deleting just send a DELETE request to the URL of the entity you want to delete.

21.11 Batch

A batch request contains multiple sub requests that will be handled in one database transaction. It is advised to use batch requests for writing operations that have any kind of relations (e.g. creating a company and an address). You can also use batch requests for reading.

A simple batch request that reads a company and all its addresses (authentication and user agent stripped for clarity):

```
curl -X POST "/api/$batch" -H "Content-Type: multipart/mixed;boundary=batch_123test" --data-binary "@data.txt"
data.txt:
--batch_123test
Content-Type: application/http
Content-Transfer-Encoding: binary

GET Crm_Company(01234567-89AB-CDEF-0123-456789ABCDEF) HTTP/1.1

--batch_123test
Content-Type: application/http
Content-Transfer-Encoding: binary

GET Crm_Address?$filter=CompanyId eq 01234567-89AB-CDEF-0123-456789ABCDEF HTTP/1.1

--batch_123test--
```

The line breaks are important. You can read more about the format in the [protocol specifications](#)

You can also read and write in one request. The sub request will be handled in their given order and the whole request is handled in one database transaction. In Appendix A you can find an example.

Writing in batch request is done in change sets. On the server the whole batch will be processed in a single database transaction which will contain sub transactions for each unique change set in the batch request. That means when a change set fails its transaction will be rolled back and no changes take place from that change set.

21.11.1 Continue on Error

By default batch execution is stopped when a request fails. If you want the server to still continue execution, you can set the header `Prefer` to `odata.continue-on-error=true`.

21.12 Return Codes

- 200 OK successful GET or PUT
- 201 CREATED successful POST
- 204 NO CONTENT successful POST or PUT with header `Prefer: return=minimal, successful DELETE`
- 400 BAD REQUEST failed POST or PUT
- 403 FORBIDDEN the user is not allowed to access (read/write) this entity type
- 404 NOT FOUND failed GET, DELETE or PUT for unknown Id
- 409 CONFLICT failed POST if entity with this id already exists
- 500 INTERNAL SERVER ERROR failed unsupported request or API error
- 501 NOT IMPLEMENTED failed PATCH

Note that there can also be other return codes. Also note, that a batch request returns 200 OK even though not all sub requests were successful.

Failed requests will be logged on the server for further analysis.

21.13 Client Development

You should not implement this yourself as there are already great libraries out there that can consume the meta data model and generate classes and handle requests and responses. You can check this as a starting point: <https://www.odata.org/libraries>

21.14 Further Reading

- <https://www.odata.org/>
- <https://docs.microsoft.com/de-de/odata/>

21.15 Known Limitations

- Aggregation Extension (groupby, distinct etc.) are not supported yet
- PATCH is not supported as multiple copies of the entity can exist in our eco system, we do not want to mix data from different sources in one entity
- ETag is not supported
- when accessing single entities by key, nested property access is not supported:
`/api/Crm_Company(95c72f4b-fc89-e511-a145-005056c00008)/StandardAddress($select=Name1)` use a `$filter` instead
- filtering and sorting in expanded collections is supported, but is done only *in memory*, meaning it is not translated to SQL, please keep that in mind when handling large data and make separate requests if possible
- not all entities are writable and also not all properties are writable, there are some indicators that reflect the limitations (see annotations above), but they may not be complete
 - you can use the `DirectSave` header to loosen that restriction in a lot of cases
- insert and update operations are single entities only, navigation properties are ignored
- self referencing models (like Company - ParentCompany / Subsidiaries) support `$expand` only on the outer level
- not all extension values are open types, especially extensions of lookups are not open types

21.16 Appendix A

The following request creates a new address entity, a new email entity referencing the address entity and then selects the newest email from the database.

```
curl -X POST "/api/$batch" -H "Content-Type: multipart/mixed;boundary=batch_123test" --data-binary "@data.txt"
```

21.16.1 Request

data.txt:

```

--batch_123test
Content-Type: multipart/mixed; boundary=changeset_456test

--changeset_456test
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 1

POST Crm_Address HTTP/1.1
Content-Type: application/json
Prefer: return=minimal

{
  "Id": "01234567-89AB-CDEF-0123-456789ABCDEF",
  "Name1": "L-mobile",
  "Name2": "solutions",
  "Name3": "GmbH & Co. KG",
  "Street": "Im Horben 7",
  "ZipCode": "71560",
  "City": "Sulzbach an der Murr",
  "Country": "Germany",
  "CountryKey": "100",
  "RegionKey": "100",
  "POBox": null,
  "ZipCodePOBox": null,
  "AddressTypeKey": "1",
  "CompanyId": "FEDCBA98-7654-3210-FEDC-BA9876543210",
  "Latitude": null,
  "Longitude": null,
  "LegacyId": null,
  "IsCompanyStandardAddress": false,
  "CreateDate": "0001-01-01T00:00:00Z",
  "ModifyDate": "0001-01-01T00:00:00Z",
  "CreateUser": null,
  "ModifyUser": null,
  "IsActive": true,
  "ExtensionValues": {
    "Distance": 1
  }
}
--changeset_456test
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 2

POST Crm_Email HTTP/1.1
Content-Type: application/json
Prefer: return=minimal

{
  "Id": "00000000-0000-0000-0000-000000000000",
  "ContactId": "FEDCBA98-7654-3210-FEDC-BA9876543210",
  "AddressId": "01234567-89AB-CDEF-0123-456789ABCDEF",
  "LegacyId": null,
  "TypeKey": "EmailWork",
  "Data": "info@l-mobile.com",
  "Comment": null,
  "CreateDate": "0001-01-01T00:00:00Z",
  "ModifyDate": "0001-01-01T00:00:00Z",
  "CreateUser": null,
  "ModifyUser": null,
  "IsActive": true
}
--changeset_456test--
--batch_123test
Content-Type: application/http
Content-Transfer-Encoding: binary

GET Crm_Email?$top=1&orderby=CreateDate desc HTTP/1.1

--batch_123test--

```

21.16.2 Response

```
--batchresponse_bf89bd68-c313-47a7-9c8d-ee9e16cbe0a3
Content-Type: multipart/mixed; boundary=changesetresponse_7ee37648-38d0-48ac-84cd-05f6017e2bbf

--changesetresponse_7ee37648-38d0-48ac-84cd-05f6017e2bbf
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 1

HTTP/1.1 204 No Content
Location: /api/Crm_Address(01234567-89ab-cdef-0123-456789abcdef)
OData-EntityId: 01234567-89ab-cdef-0123-456789abcdef

--changesetresponse_7ee37648-38d0-48ac-84cd-05f6017e2bbf
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 2

HTTP/1.1 204 No Content
Location: /api/Crm_Email(5ea64ddf-5e7a-42ab-aed0-aa080148bb6e)
OData-EntityId: 5ea64ddf-5e7a-42ab-aed0-aa080148bb6e

--changesetresponse_7ee37648-38d0-48ac-84cd-05f6017e2bbf--
--batchresponse_bf89bd68-c313-47a7-9c8d-ee9e16cbe0a3
Content-Type: application/http
Content-Transfer-Encoding: binary

HTTP/1.1 200 OK
Content-Type: application/json; odata.metadata=minimal; odata.streaming=true
OData-Version: 4.0

{
  "@odata.context": "/api/$metadata#Crm_Email",
  "value": [{
    "Id": "5ea64ddf-5e7a-42ab-aed0-aa080148bb6e",
    "ContactId": "FEDCBA98-7654-3210-FEDC-BA9876543210",
    "AddressId": "01234567-89ab-cdef-0123-456789abcdef",
    "LegacyId": null,
    "TypeKey": "EmailWork",
    "Data": "info@l-mobile.com",
    "Comment": null,
    "CreateDate": "2019-03-06T19:56:52.6290042+01:00",
    "ModifyDate": "2019-03-06T19:56:52.6290042+01:00",
    "CreateUser": "default.1",
    "ModifyUser": "default.1",
    "IsActive": true
  }]
}
--batchresponse_bf89bd68-c313-47a7-9c8d-ee9e16cbe0a3--
```


22 API

22.1 Attributes

22.1.1 ExplicitEntitySet

Use this attribute to mark REST types that have no sync controller but you still want to access them with an oData entity set.

22.1.2 ExplicitExpand

Use this attribute to mark properties that need to explicitly be expanded to be transferred via oData. Usually this is used for collections and complex types (or other navigation properties) as it has direct impact on the resulting SQL query and thus the amount of data going over the wire. If you add an entity type to a rest model (e.g. an address to a company) and it is not marked with this attribute, the resulting SQL will always load the address, but the transferred JSON will omit the address if there is no expand in the query.

22.1.3 RestrictedField

This attribute marks properties that cannot be projected into the SQL database. This means there is a complex auto map or other logic in between. Marking these properties results in the oData meta data model reflecting these restrictions, so that the client can know about them. In the meta model the properties will be marked as NotFilterable, NotCountable, NotSortable, NotExpandable, NotNavigable

```
[Database(Ignore=true)] has the same effect regarding the restrictions
```

22.1.4 NotMapped

This will mark the property to be ignored, resulting it not being listed in the meta data model.

```
[JsonIgnore] has the same effect regarding the property being ignored
```

22.2 Configuration

If attributes are not enough for you, or you need to override or extend defaults, you can implement or an `IPropertyConfigurator` for dynamic filtering and configuration.

23 API Development

23.1 Client

The (material) client currently uses two different mechanisms to build its model.

For offline mode it uses the REST models together with the relations found in JavaScript (usually `OfflineModel1.js`). For online mode it uses only `/api/$metadata` which must contain all relations (navigation properties).

23.2 Attributes

23.2.1 ExplicitEntitySet

Use this attribute to mark REST types that have no sync controller but you still want to access them with an oData entity set.

23.2.2 ExplicitExpand

Use this attribute to mark properties that need to explicitly be expanded to be transferred via oData. Usually this is used for collections and complex types (or other navigation properties) as it has direct impact on the resulting SQL query and thus the amount of data going over the wire. If you add an entity type to a rest model (e.g. an address to a company) and it is not marked with this attribute, the resulting SQL will always load the address, but the transferred JSON will omit the address if there is no expand in the query.

this attribute will add a `PreCondition` to the mapping used by Automapper, be sure to not use a `PreCondition` in your mapping of the property, as it will not be applied

23.3 RestrictedField

This attribute marks properties that cannot be projected into the SQL database. This means there is a complex auto map or other logic in between. Marking these properties results in the oData meta data model reflecting these restrictions, so that the client can know about them. In the meta model the properties will be marked as `NotFilterable`, `NotCountable`, `NotSortable`, `NotExpandable`, `NotNavigable`

`[Database(Ignore=true)]` has the same effect regarding the restrictions

23.3.1 NotMapped

This will mark the property to be ignored, resulting in it not being listed in the meta data model. Automapper will also ignore these properties.

`[JsonIgnore]` has the same effect regarding the property being ignored

23.3.2 NotReceived

This marks the property as read only.

23.3.3 RestrictedType

Marks restricted types regarding the operations that are allowed on them.

23.4 Extending the API

There are multiple interfaces which let you implement services that can extend the API easily.

23.4.1 Configuring the model

To get full access during the building phase of the model you can implement `IModelConfigurator`. Use this to register special types or operations (functions or actions). When you want to add a special controller, you can use `ODataControllerEx` as a base class. In case you are handling entities (which you almost always are), you should also implement `IEntityApiController` to get authorization checks for the entity type (meaning checking if the user has permission to access this type).

23.4.1.1 Configuring and filtering properties

If attributes are not enough for you, or you need to override or extend defaults, you can implement `IPropertyConfigurator` for dynamic filtering and configuration of properties.

23.4.1.2 Configuring types

Services implementing `ITypeConfigurator` are called a bit later (almost as last step), when all convention based exploring and registering is already done.

23.4.1.3 Annotations

To change default annotations or add your own, implement `IModelAnnotator`.

23.4.2 Manipulating queries

Sometimes you need to do extend the filtering or ordering of the query. Then you can implement `IODataQueryFunction` to manipulate the NHibernate query.

23.4.3 Manipulating results

As `ExtensionValues` are open complex types, you can add additional data to them when sending the result to the client. To do so implement `IODataDynamicExtensionValueSerializer`.

23.4.4 Manipulating data manipulating requests

You can hook into any write operations (POST, UPDATE, DELETE) by implementing `IODataWriteFunction`. It is called after the default write operation is done.

23.4.5 Manipulating keys

You can change the key of the current operation is using by implementing a `IODataAlternativeKeyProvider`. This is currently used by the OData integration to allow retrieving and writing data without knowing the L-mobile FIELD key, but only the external key.

24 OpenId configuration

24.1 Preparation

We need to have a credential to connect our application to the provider. When the credential is being created the callback url should be provided, which is *host/Account/OpenIdCallback* in our case. When creating the credential, the flow have to be defined: we suggest code flow.

24.1.0.1 What exact informations do we need?

Authority - which is the endpoint to the authentication provider. *ClientId* - identifier for the application, it is generated during creation *ClientSecret* - a password which we can use to authenticate the application against the provider *Scopes* - this will define what kind of informations we will receive from the provider, in our case **profile** is mandatory

24.2 Configuration

Once the credential is created, it is easy and straight forward: in the Main plugin's appSettings there are a few entries, such as OpenId/Authority, OpenId/ClientId, OpenId/ClientSecret, ...

24.3 FAQ

24.3.1 How does the application know which user has logged in?

When a user logs in via an identity provider, we will receive some informations about the user, including some personal informations: username, email, etc... We use the *name* property from the received claims to identify the user, and this value should be assigned to one of the users in the CRM. The assignment is very easy, you need to put this value into the *OpenId-Identifier* field.

24.3.2 Is it necessary to create a corresponding user in the CRM?

Yes. Roles, usergroups, stations, and other entities are CRM related, so we need the same user to be defined in the CRM as well with the proper settings.

24.3.3 What happens if a user logs in with the identity provider, but it's username is not assigned to any user in the CRM?

Nothing, the user will be redirected to the login page, as we can't identify this user.

25 Unit Testing with Jest

25.1 Environment

The unit tests root folder should be the Crm.Web folder, not the Crm.Test, because of resource availability, node_modules folder only available there.

The tests are running under Node.js environment with is extended with JSDom to fake browser API. There might be some complex API calls, which need polyfill to work. There are two major API-s, which already has prepared polyfill in the code, one is canvas and the other is webSql.

We are using Jest version 27, ESM still experimental, because Node.js ESM support is not quite there just yet, so the tests preferred to be written in CommonJS. This means in practical sense, you dont use import rather than use require(). (If you unsure what this mean, read after how Node.js works).

Jest has a plugin in place to compile Typescript automatically, so everything under the sun can be tested. Be aware, if you opting in for Typescript, it will be type checked!

Naming conventions:

- Name the test the same as your file what you want to test
- If you testing a general API make some resemblance in name what are you trying to test at least
- File ending **must** contain spec or test before the filetype, this is how Jest picks up what is test what is not
- File types enabled by default, js, jsx, ts, tsx

25.2 Matchers overview

Reference for all matchers, [Jest Matchers](#)

- expect(...).toBe(...), runs === check
- expect(...).toEqual(...), runs deep check, used in object
- expect(...).toHaveLength(...), calls Array.length then checks it with ===
- expect(...).toBeTruthy(), runs ==
- expect(...).toBeFalsy(), runs ==
- expect(...).not.toBe(), runs a !(...) with the chained function

25.3 How to create a unit test

1. Under the Crm.Web/Content/tests/ create your test MyNewUnitTest.spec.js.
2. Create the Test suit. You can use function() {} as well.

```
describe("MyNewUnitTest", () => {  
  
});
```

3. Create your test case inside, use the best matcher according to your needs.

```
describe("MyNewUnitTest", () => {  
  test("my creatively named test", () => {  
    const a = 2 + 2;  
    expect(a).toBe(4);  
  })  
});
```

4. Run it with `npm test .\Content\tests\MyNewUnitTest.spec.js`
 - or if using Rider, you will see a green arrow next to your test
 - if you using VS, install VSCode, and read the following [article](#)

25.4 Tests with imports

1. Import your function

1. If its using exports

```
const HelperString = require("relative/path/to/the/file").HelperString;
```

1. If its writing the window, a.k.a with side-effects

```
require("relative/path/to/the/file");
```

2. Test your code

```
test("with imported function", () => {  
  expect(importedAdder(2, 2)).toBe(4);  
  expect(window.Adder(2, 2)).toBe(4);  
})
```

25.5 Test with Promises

There is 2 main ways to do it

- Done Callback, you provide a callback on top of the function, which will terminates the test. If you forget to call it, the test will time out.

```
test("my second creative test", (done) => {  
  window.Helper.Database.initialize().then(() => {  
    expect(true).toEqual(false);  
    done()  
  })  
})
```

- Async keyword, you must use async/await in pairs!

```
test("my third creative test", async () => {  
  await window.Helper.Database.initialize()  
  expect(true).toEqual(false);  
})
```

Note, that you cannot create an `async (done) => {}` test, one or the other.