Deltek

Deltek Touch 3.6 for Maconomy® 2.5.2

Layout Configuration Guide

December 2020

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This publication contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, or translated into another language, without the prior written consent of Deltek, Inc.

This edition published December 2020.

© Deltek, Inc.

Deltek's software is also protected by copyright law and constitutes valuable confidential and proprietary information of Deltek, Inc. and its licensors. The Deltek software, and all related documentation, is provided for use only in accordance with the terms of the license agreement. Unauthorized reproduction or distribution of the program or any portion thereof could result in severe civil or criminal penalties.

All trademarks are the property of their respective owners.

Contents

Overview 1
Audience 1
Customization Tasks 2
Customization Process Using Maconomy Extender 2
Customization Process Using a Text Editor 3
Configure the Timesheet Line or Add Job Screen 5
Configure an Approval or a Summary Screen 5
Add a New Search or Lookup Field 6
Modify Search or Lookup Fields 7
Restrict Data in Basic Search Screens 7
Restrict Data in Advanced Search Screens 8
Add Related Option List fields 11
Change the Style 13
Example 13
Heavyweight Configuration 14
Query Configuration 15
Add Fields 17
Change Order 18
Change Condition 19
Layout Files 20
MScript Layout Files 20
REST Layout Files 20
Layout List Files 21
Layout File Matrix 24
Time Registration 24
Expense and Mileage Registration 24
Approvals 25
Search Layouts 27
Purchase Order Layouts 28
Absence and Allowance Requests Layouts 28
Layout Configuration 30
Customization Capabilities 31
Approval 31

Summary 31
Log Entry 32
Map 32
Layout Components and Attributes 33
Non-Editable Layout Components 33
Editable Layouts Components 33
Component Layout Attributes 34
Style Layout Attributes 39
Margin and Padding Box Model 45
Class Layout Attributes 45
CustomLayout.scss 46
Old and New Styling Methods 51
Syntax Descriptions 53
Label 53
Description 53
Rules 53
Example 53
LinkLabel 54
Description 54
Example 54
Field 54
Description 55
Rules 55
Examples 55
Date Period Field 60
Description 60
Rules 61
Examples 61
Project Lookup Field 62
Description 62
Rules 62
Examples 63
Lookup Field 64
Description 64
Rules 64
Examples 65

SearchField 66
Description 66
Rules 66
Examples 66
LinearText 67
Description 67
Rules 67
Examples 68
Grid 70
Description 70
Rules 71
Examples 72
TextField 78
Description 78
Rules 79
Examples 79
TextareaField 80
Description 80
Rules 80
Examples 80
DocumentField 81
Description 81
Rules 82
Group 83
Description 83
Rules 83
Example 83
Form 85
Description 85
Rules 85
Example 86
Attachment Icon 87
Description 88
Rules 88
Examples 88
Location Picker 88

Description 89
Rules 89
Examples 89
ReadOnlyField 90
Description 90
Rules 90
Examples 90
Conditional Function Helpers 91
Description 91
Rules 91
Example 91
Metadata Files 92
Example 92
Example 92
Appendix A: Screens, Layouts, and Metadata Files 94
Search Screens 94
Basic Search Screens 94
Advanced Search Screens 101
Approval Screens (REST based) 108
Approval Screens (MScript based) 124
Time Registration Screens 143
Expense and Mileage Registration Screens 151
Appendix B: If You Need Assistance 157
Customer Services 157
Deltek Support Center 157
Access Deltek Support Center 158
Additional Documentation 158

Overview

The Deltek Touch for Maconomy Layout Configuration Guide describes tasks necessary for the customization of layouts of the Deltek Touch for Maconomy 3.6 application using either MScript or RESTful Web services. For example, the customers may want to add or remove fields to/from a screen.

Attention: For technical information, such as pre-installation requirements and compatibility mode definition, see the *Deltek Touch for Maconomy Release Notes*.

This document contains the following topics:

- Layout Configuration Introduction
- Customization Tasks
- Layout Components and Attributes
- Syntax Descriptions
- Screens, Layout Files, and Metadata Files

There are topics, however, that only apply to either MScript or REST.

Note: The official name of the application is *Deltek Touch for Maconomy*. This document only uses it at first mention. The succeeding instances of the application name display *Deltek Touch*. In addition, the application name in *Apple App Store* and *Google Play Store* displays *Deltek Touch for Maconomy*.

Audience

The guide is intended for the technical consultant or power users who make layout changes to Deltek Touch. This document explains only common configuration tasks and field attributes.

Note: Once you have configured the system, you must log on again to Deltek Touch to ensure that all changes take effect. Other than this, there is no need to restart anything (for example, the Coupling Service).

Customization Tasks

This chapter discusses several customization tasks that you can perform on Deltek Touch screens as well as the customization process itself.

You can now customize Deltek Touch by using Maconomy Extender 1.5.3 or later.

Using Extender for customizations is highly recommended, since it allows easy deployment, user friendly editor and source control for your customizations.

Customization Process Using Maconomy Extender

Maconomy Extender 1.5.3 and later supports Touch customizations, including the following items:

- **Layout Configuration** – This allows changes to screen layouts (such as adding, removing and reordering fields on screens). In your Extender project, create a Custom folder with the Layouts sub-folder in the Touch directory. Drag and copy the standard layout file into the said folder, and make necessary changes to the file.
- **Server-Side Settings** – This allows changes to settings relevant for the business functionality in Deltek Touch (such as submit mode, week start, SSO, and custom submit timesheet action). In your Extender project, create a Custom folder in the Touch directory, if it does not yet exist. Drag and copy the standard DeltekTouch.I file into the said folder, and change the settings.
- **Search Configuration** – This allows changes to queries used for search screens (such as Task List and Find Job). In your Extender project, create a Custom folder in the Touch directory, if it does not yet exist. Drag and copy the standard DeltekTouch.I file into the said folder, and change the search layouts and/or queries.
- **Query Configuration** – *This item only applies to MScript.* This allows changes to queries used for list screens (such as Expense Sheets, Rejected Timesheets, and Timesheets to Approve). In your Extender project, create a Custom folder in the Touch directory, if it does not yet exist. Drag and copy the standard DeltekTouch.I file into the said folder, and change the search layouts and/or queries.
- **Maconomy CGI Settings** – *This item only applies to MScript.* This allows changes to the MaconomyTouch CGI program used for Touch Web service (such as, timeout and logging). In your Extender project, create a server-specific mapping file. Drag the standard MaconomyTouch.I into the said file, and change the settings. Make sure that the mappings are set properly so that the file is deployed on the correct Maconomy Web server.

Attention: For additional details, check out the *Configuration Guide and Extender Handbook*.

Customization Process Using a Text Editor

In case you do not have access to Extender, you can still customize Touch using any text editor.

Note: All layout files are saved in the **MaconomyMScript/Standard/Layout** folder.

Before you change any of these settings, Deltek recommends that you make a copy of the DeltekTouch.I file in the MaconomyMScript\Standard folder, and place it in the MaconomyMScript\Custom folder. Then change the file in the Custom folder instead of the original file.

Remember to remove the licensing information from the top when changing the .I files. To apply your changes, you need to log on again to Touch; there is no need to restart anything else.

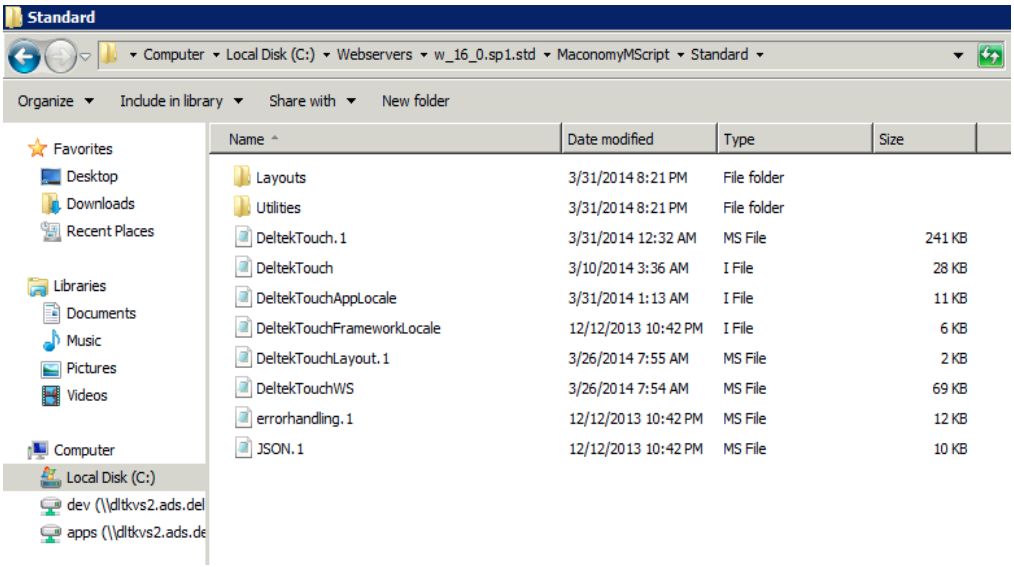
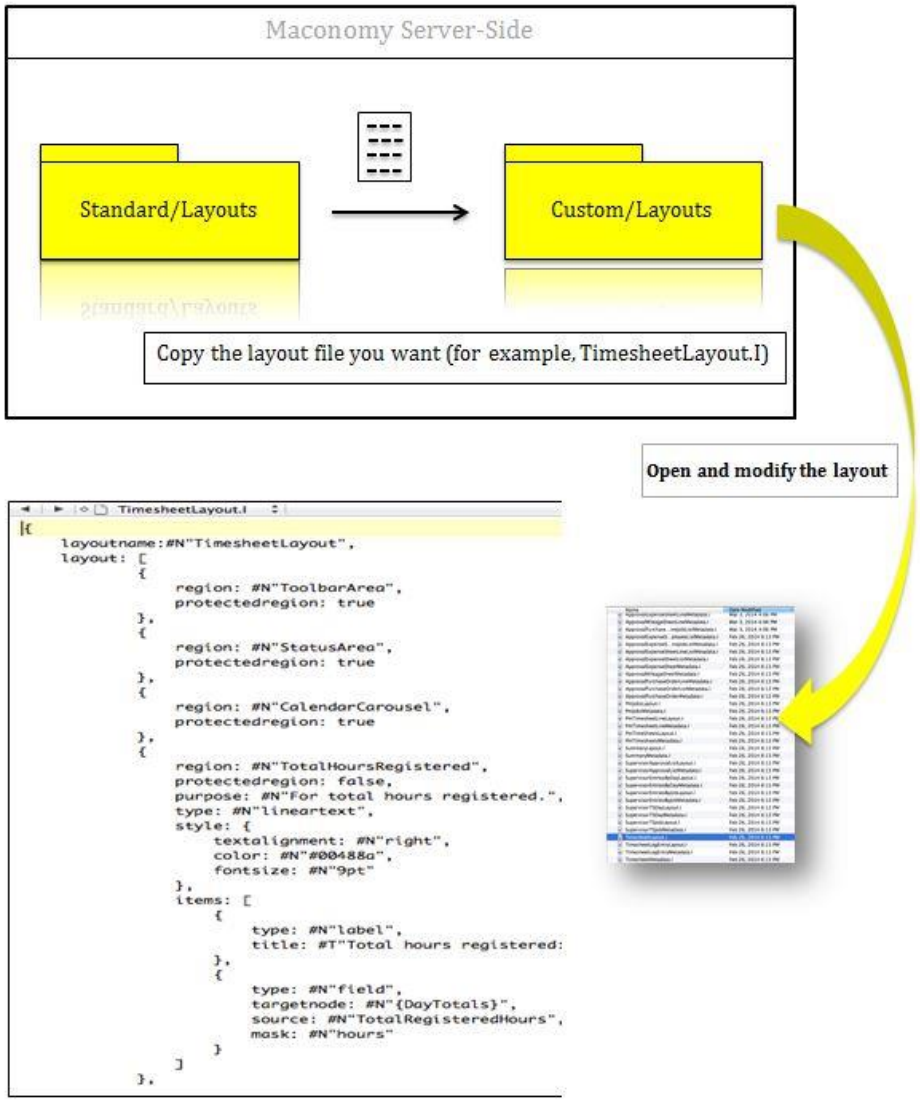
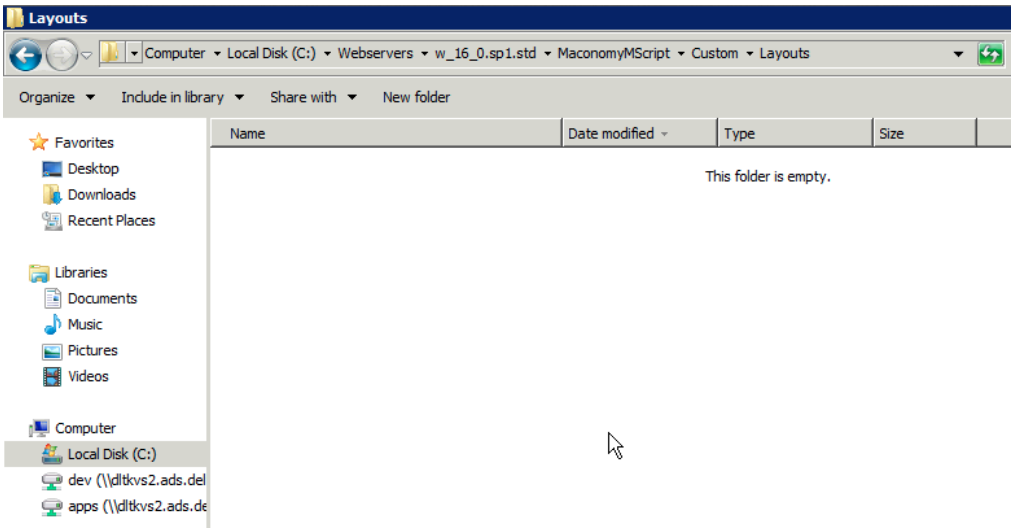


Figure 1: General Process Flow Diagram



1. From the **Standard/Layout** folder, copy the layout file you want to modify to the **Custom/Layout** folder. Creating a copy of the file ensures that the customization is preserved when the customer upgrades the system.

   Note: By default, the **Custom/Layouts** folder does not exist upon installation. You need to create it manually in the **MaconomyMScript** folder. The **Custom** folder displays on the same level as the **Standard** folder.



2. Open the layout file and make the necessary changes.

   Note: Deltek recommends that you give careful consideration to the syntax, declaration, and rules.

3. When you are done, save the layout.
4. Log in again to Deltek Touch to validate if the customization has already been applied to the screen.

Configure the Timesheet Line or Add Job Screen

To configure the Timesheet Line or Add Job screen, complete the following steps:

1. In the **MaconomyMScript** folder, create a new folder called **Custom**. The **Custom** folder displays on the same level as the **Standard** folder.
2. Create a Layouts folder under the Custom folder
3. Copy **TimesheetLogEntryLayout.I** from **Standard/Layouts** to **Custom/Layouts**.
4. In the **Custom** folder, edit **TimesheetLogEntryLayout.I** in any text editor.
   - Make sure that you remove license lines. If present, these lines display on the top.
   - Change the file accordingly to your needs

Configure an Approval or a Summary Screen

To configure the Approval or Summary screen, complete the following steps:

1. In the **MaconomyMScript** folder, create a new folder called **Custom**. The **Custom** folder displays on the same level as the **Standard** folder.
2. Copy the particular layout file that you need to customize from **Standard** to **Custom**.
3. In the **Custom** folder, edit the layout file in any text editor.
4. If you want to add or update the fields in the layout file, you can use the corresponding metadata file of this layout file. It uses the file name format **[layoutfilename]MetaData**, for example, TimesheetMetadata.I.

Attention: For the list of layouts and corresponding metadata files, see the Appendix A: Screens, Layouts, and Metadata Files section of this document.

Add a New Search or Lookup Field

To add a lookup type field to a REST layout, complete the following steps:

Note: For example, add **Specification3Name** to **TimesheetLogEntryWeeklyLayout.REST.I**.

1. Add the LayoutList_REST.I in to the **/Custom/Layouts** folder.
2. Modify the LayoutList_REST.I and add the new lookup layout. For example, Find_Specification3.I, find_specification3.
3. When creating a new lookup layout, the layoutname should be the same with the containerName found on the specification's relatedContainer. This layout name is case-sensitive. For example, layoutname: #N"find_specification3."
4. Use the custom lookup layout by adding the following code in to the TimesheetLogEntryWeeklyLayout_REST.I. In adding the field specification3 to this layout, searchlayoutname should be the same with what you set with LayoutList_REST.I.

```
{       label: #T”Location of Work”,
          source: #N”Specification3Name”,
        searchlayout: #N"Find_Specification3",
        type: #N"lookupfield",
          queryfieldname: #N"Specification3Name",
        displayfields:  [
        {
             source: #N"Specification3Name",
            queryfieldname: #N"Specification3Name"
        },
         {
             label: #N" - "
         },
         {
             source: #N"Specification3DescriptionVar",
             queryfieldname: #N"Description"
          }
        ]
},
```

Modify Search or Lookup Fields

This procedure allows you to restrict the data shown in basic and advanced search screens.

Attention: For the list of layouts, see the Appendix A: Screens, Layouts, and Metadata Files section of this document.

Restrict Data in Basic Search Screens

Basic search screens include screens like **Find Activity**, and **Find Task**. All basic search screens behave similarly.

To modify the search data for a specific field on one screen, add the restrictions to the screen layout. However, to modify the search data for a specific field on multiple screens, add the restrictions to the corresponding search layout. You can add the restriction on the search layout, and the value is automatically appended on the screen layout. For example:

- If for timesheets you want to use only tasks in Company 1, while for expense sheets you want to use only tasks in Company 2, take the following steps:
  - Add the Company 1 restriction to the TaskName lookup field, on the Timesheet Line screen layout (TimesheetLogEntryWeeklyLayoutREST.I or TimesheetLogEntryDailyLayoutREST.I)
  - Add the Company 2 restriction to the TaskName lookup field, on the Expense Sheet Line screen layout (ExpenseSheetLineLayoutREST.I)
- If everywhere in Touch (time, expense, mileage…), you want to use only tasks in Company 1:
  - Add the Company 1 restriction to the Find Task search layout (Find_TaskListLine.I)

To customize basic search screens:

1. Copy the standard layout, either the screen or search layout, containing the field to be modified to the **/Custom/Layouts** folder.
2. Modify the custom layout to include the restriction. To append your condition, insert the following:

```
options : {
   restriction: #N"samplequery = 1"
}
```

   Modify the bolded text according to your specification. Replace samplequery with the queryfieldname you want to restrict, and replace 1 with the value you want shown on the screen.

Examples

- Restriction on the lookup field in the screen layout (TimesheetLogEntryWeeklyLayoutREST.I):

```
{
      source: #N"TaskName",
      label: #T"Task",
      searchlayoutname: #N"find_tasklistline",
      queryfieldname: #N"taskname",
      type: #N"lookupfield",
      options: {
```

```
                restriction: #N"blocked=false"
        },
        displayfields: [
        {
          source: #N"taskname",
          queryfieldname: #N"taskname"
        },
        {
          label: #N" - "
        },
        {
          source: #N"tasktextvar",
          queryfieldname: #N"description"
        }
      ]
    },
```

- Restriction in the search layout (Find_TaskListLine.l):

```
        {
                protectedregion: false,
                region: #N"SearchFieldArea",
                type: #N"searchfield",
                filterkeys: ["taskname", "description"],
                placeholder: #T"Task Number, Name"
                options: {
                        restriction: #N"blocked=false"
                        }
            },
```

Restrict Data in Advanced Search Screens

Advanced search screens include **Find Vendor**, **Find Job**, and **Address** search screens. The data is provided by two or more REST requests (filter and/or foreignkey) to different containers. You can customize the restriction on all these REST requests.

The following are the 3 advanced search screens, and the options available for customization:

1. **Find Vendor** – This screen is used on the **Vendor** field in purchase orders, and has 2 options:
   - **recentlyused** – This option refers to the REST request which retrieves the jobs used on the most recent purchase orders. When you open the screen, this request is invoked.
   - **all** – This option refers to the REST request which retrieves all jobs. When you tap the magnifying glass icon, this request is invoked.

A restriction added on an option is applied only on the REST request related to the option. Restrictions can be added in the screen layout (PurchaseOrderLayoutRest.I) and/or search layout (FindVendor_PurchaseOrder.I). If you add a restriction on the FindVendor_PurchaseOrder.I search layout, it will append its value to the "all" restriction from the screen layout. This is an option if you want to add the restriction inside the "all" option.

2. **Find Job** – This screen is used on the **Job** field on timesheet lines, expense sheets and expense sheet lines, mileage sheet and mileage sheet lines, and purchase orders and purchase order lines. This screen has the following options:

   - **recentlyused** – This option refers to the REST request which retrieves the jobs used on the most recent items (for example, timesheets). When you open the screen, this request is invoked.
   - **all** – This option refers to the REST request which retrieves all jobs. When you tap the magnifying glass icon, this request is invoked.
   - **favorite** – This option refers to the REST request which retrieves favorites. When you tap the Favorite tab, this request is invoked.
   - **myjob** – This option refers to the REST request which retrieves jobs you have access to based on employee control. This request is invoked when you open the screen, if **ShowJobsWithEmployeeControl** is set to **True** in DeltekTouch.I

   You can add any option (Recently Used, My, All, and Favorite) that is applicable in the following contexts:

   - Timesheet line options – recently used, my, all, favorite
   - Expense sheet options – recently used, all
   - Expense sheet line options – recently used, all, favorite
   - Mileage sheet options – recently used, all
   - Mileage sheet line options – recently used, all, favorite
   - Purchase order options – recently used, all
   - Purchase order line options – recently used, all

3. **Address** – The **From** and **To** address search screens are available for mileage sheet lines. You can only add the restriction on the MileageSheetLineLayoutREST.I screen layout.

To customize advanced search screens:

1. Copy the standard layout, either the screen or search layout, containing the field to be modified to the /Custom/Layouts folder.
2. Modify the custom layout to include the restriction. Take any of the following actions:

   a) Specify a default restriction, which is applied to all REST requests issued for the advanced search screen. Syntax:

   ```
   options : {
      restriction: #N"samplequery = 1"
   }
   ```

   b) Specify different restrictions for the different REST requests used for the advanced search screen. The REST requests are represented by option names (for example, all). The option names (option1, option2) are specific to the advanced search screens. Syntax:

   ```
   options: {
   ```

```
option1: {
   restriction: #N"samplequery = 1"
},
option2: {
   restriction: #N"samplequery = 2"
}
```

Modify the bolded texts according to your specification. Replace samplequery with the queryfieldname you want to restrict, and replace 1 with the value you want shown on the screen.

Examples

- Restriction on the **Vendor** lookup field in the screen layout (PurchaseOrderLayoutREST.l):

```
{
   label: #T"Vendor",
   source: #N"suppliernumber",
   queryfieldnames: [#N"vendornumber", #N"suppliernumber"],
   type: #N"vendorlookupfield",
   mode: #N"project",
   searchlayoutname: #N"FindVendor_PurchaseOrder",
   options: {
         recentlyused: {
               restriction: #N"AllowPurchaseOrders = 0"
         },
         all: {
               restriction: #N"vendorgroup = VendorGroupType'europe'"
         }
   },
   displayfields: [
   {
         source: #N"suppliernumber",
         queryfieldnames: [#N"vendornumber", #N"suppliernumber"]
   },
   {
         label: #N" - "
   },
   {
         source: #N"name1",
         queryfieldname: #N"name1"
   }
   ]
```

```
},
```

- Restriction in the Job search layout (Find_JobHeader_Timesheet.I):

```
{
        protectedregion: false,
        region: #N"SearchFieldArea",
        type: #N"searchfield",
        filterkeys: ["jobnumber", "jobname", "name1"],
        placeholder: #T"Job Number, Name, Customer Name",
        options: {
                recentlyused: {
                        restriction: #N"jobnumber = 'MTJ03'"
                },
                myjob: {
                        restriction: #N"jobnumber = 'MTJ03'"
                },
                all: {
                        restriction: #N"companynumber={api_currentemployee}.{companynumber}"
                }
                }
},
```

- Restriction on the **From** and **To** fields in the screen layout (MileageSheetLineLayoutREST.I):

```
{
   source: #N"MileageFrom",
   container: #N"find_recentlyusedmileagefromu",
   label: #T"From",
   type: #N"locationpicker",
   options: {
      restriction: #N"sample=1"
   }
}
```

Add Related Option List fields

To add option list type fields, which depend on each other, to a REST layout, complete the following steps:

> **Note:** For example, Add **State (**SelectedOption1) and **County** (SelectedOption2) to **TimesheetLogEntryWeeklyLayout.REST.I**. Values display in **County** depend on the value selected in **State**.

1. Add the LayoutList_REST.I file into the **/Custom/Layouts** folder.

2. Modify LayoutList_REST.I and add the new lookup layout. For example,

   `Find_TheOption.I, find_theoption`

3. When creating a new lookup layout, the layoutname should be the same as the containerName found in the specification's relatedContainer. This layout name is case-sensitive. For example, layoutname: #N"find_theoption"

4. Use the custom lookup layout in TimesheetLogEntryWeeklyLayout_REST.I. When adding the option listsfields to this layout, searchlayoutname should be the same as that of what you set with LayoutList_REST.I.

   For adding State (=SelectedOption1), you need to add to TimesheetLogEntryWeeklyLayout_REST.I:

```
{
  label: #T"State",
  source: #N"selectedoption1",
  type: #N"lookupfield",
searchlayoutname: #N"find_theoption",
  queryfieldname: #N"name",
  displayfields: [
    {
      source: #N"selectedoption1",
      queryfieldname: #N"name"
    }
  ]
},
```

   For adding County (=SelectedOption2), you need to add:

```
{
  label: #T"County",
  source: #N"selectedoption2",
  type: #N"lookupfield",
searchlayoutname: #N"find_theoption",
  queryfieldname: #N"name",
restriction: #N"remark1 = selectedoption1",
  displayfields: [
    {
      source: #N"selectedoption2",
      queryfieldname: #N"name"
    }
  ]
},
```

Change the Style

This procedure allows you to change the existing style.

To change the style, complete the following steps:

1. Go to the Custom folder.
2. Add the CssClass.I file.
3. Check the `className` name you want to modify.
4. If the layoutList_Mscript file does not yet exist in the custom folder, copy the file from the standard folder.
5. In LayoutList_MScript.I, include `CssClass.I, CssClass` in the LayoutList_MScript.I file.

> **Note:** Make sure that you place the ClssClass.I file in the first line of the LayoutList_MScript.I file.

Example

Below is the sample code in updating changing the color:

```
{CssClass:[
        {
                className: #N"text_medium_bold",
                style: {
                        fontsize: #N"16px!important",
                        textalignment: #N"left",
                        color: #N"#f2b600"
                }
        }
]}
```

Heavyweight Configuration

If you need to make changes to the business processes supported by Touch, we have what we call heavyweight configuration; meaning, changes to the MScript source code.

Depending on the nature of your requirements, you need to change the following files:

- **Time.1.ms** — For changes to the time registration workflow (open timesheet, save timesheet line, copy timesheet line, and submit timesheet)
- **DeltekTouch.1.ms** — For changes to the approval, expense, and mileage flows
- **DeltekTouchWS.ms** — For login changes

Regardless of the nature of your requirements, you should always create a custom version of the file you change and place it in the …/MacomyMScript/Custom folder.

The most typical heavyweight configurations are used for the following tasks:

- Saving timesheet lines, creating a new line, editing existing line
- Submitting timesheets
- Showing timesheets

In these cases, you need to perform the following steps:

1. Create MaconomyMScript/Custom folder (if it does not yet exist).
2. Copy MaconomyMScript/Standard/DeltekTouchWS.ms to MaconomyMScript/Custom folder.
3. Change MaconomyMScript/Custom/DeltekTouchWS.ms to have the proper file links:
   - uses Standard::errorhandling(1) as errorhandler;
   - uses Standard::JSON(1) as JSON;
   - uses Standard::DeltekTouch(1) as DeltekTouch;
   - uses Standard::Utilities::DateCalculation(1) as dateCal;
   - uses Standard::DeltekTouchLayout(1) as DeltekTouchLayout;
   - uses .::Time(1) as Time;
4. Change configuration.ini to use Custom/DeltekTouchWS.ms instead of Standard/DeltekTouchWS.ms, in the URL.
5. Copy Time.1.ms from Standard to Custom.
6. Change Custom/Time.1.ms to have the proper file links:
   - uses Standard::DeltekTouch(1) as DeltekTouch;
   - uses Standard::Utilities::DateCalculation(1) as dateCal;
7. Change Custom/Time.1.ms based on your requirements.

Note: When upgrading Touch, you need to update all your custom files manually.

Query Configuration

This section only applies to MScript.

In Touch 1.4, we opened the relevant queries used in Touch for customization to help you easily control which fields you would like in lists, condition to be used for fetching the content of the list, and sort order. This is achieved by adding so called “queries” to DeltekTouch.I.

The following screens are using queries:

- Timesheet Approval
- Timesheet Line Approval
- Job (opened from Timesheet Line Approval)
- Expense Sheet Approval
- Expense Line Approval
- Expense Sheet Lines (opened from Expense Line Approval)
- PO Approval
- PO Line Approval
- POs job (opened from PO Line Approval)
- Vendor Invoice Approval
- Invoice Allocation Approval
- Vendor Invoices (opened from Invoice Allocation Approval)
- Draft Invoice Approval
- Items for Approval
- Rejected Timesheets
- Expense Sheets (in Maconomy 2.0 and later)
- Mileage Sheets (in Maconomy 2.0 and later)

You can now perform the following actions on the mentioned screens:

- Add more fields to the layout
- Change the ordering of the items in the list
- Change the condition used to fetch the items in the list

Note: Each screen displays a list of items. These items are fetched using a query (typically to a universe; sometimes directly to the database). These queries are now available for customizations in DeltekTouch.I setup file.

For the abovementioned changes, you need to change the corresponding query in your custom Touch setup file (…\Custom\DeltekTouch.I). A query has three parts to which you can make changes:

- MSELECT– This part allows you to add fields.
- WHERE–This part allows you to change the condition for the data
- ORDER BY –This part allows you to change the ordering of the items in the list.

Note: In general, Deltek recommends that you do not delete information from the queries.

Query name

Layout file name

Metadata file name

Example of query in DeltekTouch.I:

```
// List of time sheets to be approved
SupervisorApprovalListQuery:
{
  Operation: #N"getTimesheetsToApprove",
  Layout: #N"SupervisorApprovalListLayout",
  Metadata: #N"SupervisorApprovalListMetadata",
  Title: #T"Timesheets To Approve",
  Fields:
    [
      {Name: #K"EmployeeNumber", Title: #T"Employee Number"},
      {Name: #K"EmployeeType", Title: #T"Employee Type"}
    ],
  searchInfo:
    {
    mqlQuery: 'MQL',
    MSELECT
      TimeSheetHeader.EmployeeNumber as EmployeeNumber,
      TimeSheetHeader.EmployeeName as EmployeeName,
      TimeSheetHeader.PeriodStart as StartDate,
      TimeSheetHeader.PeriodEnd as EndDate,
      TimeSheetHeader.WeekNumber as WeekNumber,
      TimeSheetHeader.TheYear as Year,
      TimeSheetHeader.Part as Part,
      TimeSheetHeader.InstanceKey as TimeSheetHeaderInstanceKey
      TimeSheetHeader.ApprovalGroupInstanceKey as ApprovalGroupInstanceKey,
      TimeSheetHeader.ExternalTimeWeekTotal as ExternalTimeWeekTotal,
```

Add fields here

```
      TimeSheetHeader.InternalTimeWeekTotal as InternalTimeWeekTotal
FROM
     Integration::MaconomyMobile::Universes::TimeSheetHeaderApprovalU
WHERE
     ApprovalLine.Approver = paramEmployeeNumber
     or (ApprovalLine.ApproverEmployeeType = paramEmployeeType
       and ApprovalLine.ApproverEmployeeType <> EmployeetypeType'Nil)
ORDER BY
      EmployeeName, StartDate
USING PARAMETERS
      paramEmployeeNumber: string,
      paramEmployeeType: EmployeetypeType
MQL
  parameters:
     [
      { Name: #K"paramEmployeeNumber", fromField: #K"EmployeeNumber"},
      { Name: #K"paramEmployeeType", fromField: #K"EmployeeType" }
     ]
  }
},
```

Add conditions here

Change ordering here

You must NOT modify the parts highlighted in **red**. You can make changes to the parts highlighted in **green**, as described in the succeeding sections.

Add Fields

If you want to add fields to the any of the screens mentioned in the previous section, you need to change your custom DeltekTouch.I before changing the corresponding layout file. To do this, complete the following steps:

1. Identify the screen which you would like to change (for example, Timesheet Approval)
2. Identify the field you would like to add (for example, the company = CompanyNumber field)
3. Find the custom layout to change (for example, …\Custom\Layouts\SupervisorApprovalListLayout.I)
4. Add field to the custom layout (CompanyNumber field to SupervisorApprovalListLayout.I)
5. Find the query to change in …\Custom\DeltekTouch.I (for example, SupervisorApprovalListQuery).
6. The easiest is to search for the layout name (for example, SupervisorApprovalListLayout) in the file.
7. Add the field to the query, after MSELECT.

   Example:
   `MSELECT`

```
        TimeSheetHeader.CompanyNumber as CompanyNumber,
        TimeSheetHeader.EmployeeNumber as EmployeeNumber,
        TimeSheetHeader.EmployeeName as EmployeeName,
        TimeSheetHeader.PeriodStart as StartDate,
        TimeSheetHeader.PeriodEnd as EndDate,
        TimeSheetHeader.WeekNumber as WeekNumber,
        TimeSheetHeader.TheYear as Year,
        TimeSheetHeader.Part as Part,
        TimeSheetHeader.InstanceKey as TimeSheetHeaderInstanceKey
        TimeSheetHeader.ApprovalGroupInstanceKey as ApprovalGroupInstanceKey,
        TimeSheetHeader.ExternalTimeWeekTotal as ExternalTimeWeekTotal,
        TimeSheetHeader.InternalTimeWeekTotal as InternalTimeWeekTotal
```

Change Order

Some of the screens allow you to change the order of the items in the list by changing your custom DeltekTouch.I.

1. Identify the screen which you would like to change (for examples, Timesheet Approval)
2. Identify the new ordering (for example, by StartDate)
3. Find the query to change in …\Custom\DeltekTouch.I (for example, SupervisorApprovalListQuery).
4. Change the order in the query, after ORDER BY.

Example:

```
ORDER BY
        StartDate
```

You must NOT change the order of items on the following screens:

- **Expense Sheet Approval** (ApprovalExpenseSheetListQuery)
- **Expense Sheet Lines**, which opened from **Expense Line Approval** (ApprovalExpenseSheetLineListQuery)
- **Vendor Invoices**, which are opened from Invoice **Allocation Approval** (ApprovalVendorInvoiceListPMQuery)
- POs job, which are opened from **PO Line Approval** (ApprovalPurchaseOrderPMListQuery)

Change Condition

For those screens that you can modify, if you would pick the items in the list using a different condition, change your custom DeltekTouch.I.

Sample Scenarios:

Users want to see only items that they could approve by themselves, not by employee type (which was the functionality in Touch 1.3).

Users would also like to view items that can be approved by **All**.

1. Identify the screen which you would like to change (for example, Timesheet Approval)
2. Identify the new condition (for example, approvals only by employee)
3. Find the query to change in …\Custom\DeltekTouch.I (for example, SupervisorApprovalListQuery).
4. Change the condition in the query, after WHERE.

Example:

```
WHERE
ApprovalLine.Approver = paramEmployeeNumber
```

Note: Changing condition is a difficult customization and must be treated with caution. Changes to conditions must be carried out across all approval queries, including the ItemsForApprovalQuery for the badges.

Layout Files

In Touch 2.0, we have two sets of layouts that correspond to the different Web services (MScript and REST) that Deltek Touch supports.

We needed the second set of layouts because the data structures provided by the two Web services are different.

The syntax and layout structure are the same for all layouts. The only difference is in the data. In some cases, the REST layout is a simple copy of the MScript layout with only the layout name and layout file name being different.

For a given area (for example, time, expense, and approvals), Deltek Touch always uses only one Web service. Therefore, once you identify which Web service you are going to use, you know which the set of layouts you need to customize.

For example:

Assuming that you use REST for time and expense, and MScript for approvals: if you need to customize time or expense layouts, you should change only the REST layout files. There is no need to customize the MScript layout files for time or expense because they are not going to be used. As for approvals, you need to customize only the MScript layout files.

MScript Layout Files

The Touch 2.0 MScript layout files are the same as the layout files in Touch 1.x.

These file have not changed since Touch 1.6.1 except for a minor change in ExpenseSheetLayout.I. This means that, if you have Touch 1.6.1 API and you customized any other layout than ExpenseSheetLayout.I, you do not need to update your custom layouts when you upgrade to Touch 2.0 (assuming that you continue using the MScript Web service).

REST Layout Files

The Touch 2.0 REST layout files are new files that were not present in Touch 1.x. The REST layout files are named similarly to the MScript layout files. The only difference is that the REST layout files are appended with “REST”.

For example:

MScript	REST
ExpenseSheetLayout.I	ExpenseSheetLayoutREST.I
TimesheetLayout.I	TimesheetLayoutREST.I

As mentioned above, in some cases, the REST layout file is almost an identical copy of the MScript layout file but with different name. There are some cases, however, when the data in the REST layout files (for example, source) is different. In addition, there are some instances when there are two REST layout files that correspond to one MScript layout files (for example, for the **Timesheet Line** screen, we have different REST layouts for the daily and weekly modes).

There are some REST layout files that have no MScript correspondent. These are mostly for search screens (for example, Find_JobHeader_Timesheet.I). In this case, we do not append the file name with REST.

Overall, the REST layout files support all configuration capabilities available in MScript layout files, allow you to perform the following tasks:

- Customize search layouts (for example: Find Job, Task List, and Mileage From)
- Have different layouts for weekly and daily timesheets
- Customize more in some screens (for example: Weeks, Calendar, and Summary)

Layout List Files

Before Deltek Touch 2.2, the list of the layouts used by Touch users was stored in DeltekTouchWS.ms.

For example:

```
// Returns the screen layouts
function getScreenLayouts()
{
var CustomLayout = {};
CustomLayout = [];
new CustomLayout[0] = DeltekTouchLayout::getLayout("TimesheetLayout.I","timesheetLayout");
new CustomLayout[1] = DeltekTouchLayout::getLayout("SummaryLayout.I","summaryLayout");
new CustomLayout[2] = DeltekTouchLayout::getLayout("TimesheetLogEntryLayout.I",
….
new CustomLayout[77] = DeltekTouchLayout::getLayout("RejectionNotificationListLayoutREST.I",
"RejectionNotificationListLayoutREST");


return CustomLayout;
}
```

In Deltek Touch 2.2 and higher, the list of layouts has been removed from DeltekTouchWS.ms. Instead, Deltek Touch has text (.I) files that store the same information:

- **LayoutList_MScript.I**: includes the list of layouts used by the MScript Touch code in alphabetical order
- **LayoutList_REST.I**: includes the list of layouts used by the REST Touch code in alphabetical order

Note: The layout list files and layouts are saved in the same location.

The released Deltek Touch product includes standard layouts and corresponding standard layout list files:

hTime ▸ 2.2 ▸ 20.0 ▸ MaconomyMScript ▸ Standard ▸ Layouts

Name	Date modified	Type	Size
Find_LocationFromTo.I	21-03-2017 16:41	I File	4 KB
Find_PurchaseOrderLine_ExpenseSheet.I	21-03-2017 16:41	I File	3 KB
Find_PurchaseOrderLine_TimeSheet.I	21-03-2017 16:41	I File	3 KB
Find_TaskListLine.I	21-03-2017 16:41	I File	2 KB
Find_Vendor.I	30-03-2017 14:13	I File	2 KB
FindJobHeader_Expense.I	30-03-2017 14:14	I File	2 KB
FindJobHeader_PurchaseOrder.I	11-04-2017 18:39	I File	2 KB
FindVendor_PurchaseOrder.I	26-04-2017 10:07	I File	2 KB
ItemsForApprovalLayout.I	19-05-2017 12:43	I File	3 KB
ItemsForApprovalMetadata.I	21-03-2017 16:41	I File	1 KB
LayoutList_MScript.I	01-06-2017 15:05	I File	3 KB
LayoutList_REST.I	01-06-2017 15:05	I File	3 KB
MileageSheetLayout.I	21-03-2017 16:41	I File	6 KB
MileageSheetLayoutREST.I	30-05-2017 11:53	I File	8 KB
MileageSheetLineLayout.I	21-03-2017 16:41	I File	4 KB
MileageSheetLineLayoutREST.I	21-03-2017 16:41	I File	6 KB
MileageSheetLineMetadata.I	21-03-2017 16:41	I File	7 KB
MileageSheetMetadata.I	21-03-2017 16:41	I File	12 KB
MileageSheetsLayout.I	21-03-2017 16:41	I File	3 KB
MileageSheetsLayoutREST.I	21-03-2017 16:41	I File	3 KB
MileageSheetsMetadata.I	21-03-2017 16:41	I File	1 KB
PmJobsLayout.I	21-03-2017 16:41	I File	2 KB
PmJobsMetadata.I	21-03-2017 16:41	I File	1 KB
PmTimesheetLineLayout.I	21-03-2017 16:41	I File	5 KB
PmTimesheetLineMetadata.I	21-03-2017 16:41	I File	2 KB
PmTimesheetLogEntryLayout.I	21-03-2017 16:41	I File	3 KB
PmTimesheetLogEntryMetadata.I	21-03-2017 16:41	I File	7 KB
PmTimeSheetsLayout.I	21-03-2017 16:41	I File	4 KB
PmTimesheetsMetadata.I	21-03-2017 16:41	I File	2 KB
PurchaseOrderLayoutREST.I	30-05-2017 11:53	I File	8 KB

The layout list file is a simple list of layout file name – layout key pairs.

Setup.txt | VitalInfo.txt | PerlScripts | LayoutList_REST.I

```
AbsenceApprovalFormLayoutREST.I, AbsenceApprovalFormLayoutREST
AbsenceApprovalListLayoutREST.I, AbsenceApprovalListLayoutREST
AbsenceRequestLayoutREST.I, AbsenceRequestLayoutREST
AbsenceRequestsLayoutREST.I, AbsenceRequestsLayoutREST
AllowanceApprovalFormLayoutREST.I, AllowanceApprovalFormLayoutREST
AllowanceApprovalListLayoutREST.I, AllowanceApprovalListLayoutREST
AllowanceApproveInputScreenREST.I, AllowanceApproveInputScreenREST
AllowanceRequestLayoutREST.I, AllowanceRequestLayoutREST
AllowanceRequestsLayoutREST.I, AllowanceRequestsLayoutREST
DocumentArchiveListREST.I, DocumentArchiveListREST
ExpenseSheetLayoutREST.I, expenseSheetLayoutREST
ExpenseSheetLineLayoutREST.I, expenseSheetLineLayoutREST
ExpenseSheetsLayoutREST.I, expenseSheetsLayoutREST
Find_AbsenceType.I, Find_AbsenceType
Find_Account.I, find_account
Find_Activity.I, find_activity
Find_Employee.I, find_employee
Find_JobFavorite_Timesheet.I, find_jobfavorite_timesheet
Find_JobHeader_Timesheet.I, find_jobheader_timesheet
Find_LocationFromTo.I, find_locationfromto
Find_PurchaseOrderLine_ExpenseSheet.I, find_purchaseorderline_expensesheet
Find_PurchaseOrderLine_TimeSheet.I, find_purchaseorderline_timesheet
Find_TaskListLine.I, find_tasklistline
FindJobHeader_Expense.I, findjobheader_expense
FindJobHeader_PurchaseOrder.I, FindJobHeader_PurchaseOrder
FindVendor_PurchaseOrder.I, FindVendor_PurchaseOrder
MileageSheetLayoutREST.I, mileageSheetLayoutREST
MileageSheetLineLayoutREST.I, mileageSheetLineLayoutREST
MileageSheetsLayoutREST.I, mileageSheetsLayoutREST
PurchaseOrderLayoutREST.I, PurchaseOrderLayoutREST
PurchaseOrderLineLayoutREST.I, PurchaseOrdersLineLayoutREST
PurchaseOrdersLayoutREST.I, PurchaseOrdersLayoutREST
QuickCaptureAttachReceiptLayoutREST.I, quickCaptureAttachReceiptLayoutREST
QuickCaptureExpenseSheetLineLayoutREST.I, quickCaptureExpenseSheetLineLayoutREST
RejectedTimesheetsLayoutREST.I, rejectedTimesheetsLayoutREST
RejectionNotificationListLayoutREST.I, RejectionNotificationListLayoutREST
SummaryDailyLayoutREST.I, SummaryDailyLayoutREST
SummaryLayoutREST.I, SummaryLayoutREST
TimeSheetCalendarLayoutREST.I, TimeSheetCalendarLayoutREST
TimesheetDaysLayoutREST.I, TimesheetDaysLayoutREST
TimesheetLayoutREST.I, TimesheetLayoutREST
TimesheetLogEntryDailyLayoutREST.I, TimesheetLogEntryDailyLayoutREST
TimesheetLogEntryWeeklyLayoutREST.I, TimesheetLogEntryWeeklyLayoutREST
TimeSheetWeekLayoutREST.I, TimeSheetWeekLayoutREST
```

If you need to add a new layout (for example, search layout), add the corresponding pair values to the list.

Note: You must NOT change the standard files. Instead, you should copy the files to the Custom folder (for example, **/MaconomyMScript/Custom/Layouts/Layoutlist_MScript.I**).

Layout File Matrix

Time Registration

MScript	REST
TimesheetLayout.l	TimesheetLayoutREST.l TimesheetDaysLayoutREST.l
TimesheetLogEntryLayout.l	TimesheetLogEntryWeeklyLayoutREST.l TimesheetLogEntryDailyLayoutREST.l
SummaryLayout.l	SummaryLayoutREST.l SummaryDailyLayoutREST.l
RejectedTimesheetsLayout.l	RejectedTimesheetsLayoutREST.l
-	TimeSheetWeekLayoutREST.l
-	TimeSheetCalendarLayoutREST.l
-	Find_JobHeader.l
-	Find_TaskListLine.l
-	Find_JobFavorite_Timesheet.
-	Find_JobHeader_Timesheet.l
-	Find_Activity.l

Expense and Mileage Registration

MScript	REST
ExpenseSheetsLayout.l	ExpenseSheetsLayoutREST.l
ExpenseSheetLayout.l	ExpenseSheetLayoutREST.l
ExpenseSheetLineLayout.l	ExpenseSheetLineLayoutREST.l
CreateExpenseSheetLineLayout.l	QuickCaptureExpenseSheetLineLayoutREST.l
QuickCaptureAttachReceiptLayout.l	QuickCaptureAttachReceiptLayoutREST.l
MileageSheetsLayout.l	MileageSheetsLayoutREST.l
MileageSheetLayout.l	MileageSheetLayoutREST.l

MScript	REST
MileageSheetLineLayout.I	MileageSheetLineLayoutREST.I
-	Find_LocationFromTo.I
-	FindJobHeader_Expense.I
-	DocumentArchiveListREST.I
-	Find_FinanceVATCode.I
-	Find_TheOption_ExpenseJustification.I

Approvals

MScript	REST
PmTimesheetLogEntryLayout.I	PmTimesheetLogEntryLayoutREST.I
PmJobsLayout.I	-
PmTimeSheetsLayout.I	-
PmTimesheetLineLayout.I	-
SupervisorApprovalListLayout.I	-
SupervisorTimesheetLogEntryLayout.I	SupervisorTimesheetLogEntryLayoutREST.I
SupervisorTSJobLayout.I	-
SupervisorTSDayLayout.I	-
SupervisorEntriesByJobLayout.I	-
SupervisorEntriesByDayLayout.I	-
ApprovalExpenseSheetListLayout.I	-
ApprovalMileageSheetLayout.I	-
ApprovalExpenseSheetLineListLayout.I	-
ApprovalExpenseSheetLayout.I	-
ApprovalExpenseSheetLineJobListLayout.I	-
ApprovalExpenseSheetLineLayout.I	-

MScript	REST
ApprovalExpenseSheetLineEmployeeListLayout.I	-
ApprovalMileageSheetLineLayout.I	-
ApprovalPurchaseOrderLayout.I	ApprovalPurchaseOrderLayoutREST.I
ApprovalPurchaseOrderJobListLayout.I	-
ApprovalPurchaseOrderListLayout.I	-
ApprovalPurchaseOrderLineLayout.I	ApprovalPurchaseOrderLineLayoutREST.I
ApprovalPurchaseOrderPMLayout.I	ApprovalPurchaseOrderPMLayoutREST.I
ApprovalPurchaseOrderLinePMLayout.I	ApprovalPurchaseOrderLinePMLayoutREST.I
ApprovalVendorInvoiceListLayout.I	ApprovalVendorInvoiceListLayoutREST.I
ApprovalVendorInvoiceLayout.I	ApprovalVendorInvoiceLayoutREST.I
ApprovalVendorInvoiceAllocationLineLayout.I	ApprovalVendorInvoiceAllocationLineLayoutREST.I
ApprovalVendorInvoiceAllocationLinePMLayout.I	ApprovalVendorInvoiceAllocationLinePMLayoutREST.I
ApprovalVendorInvoiceAllocationLineJobListLayout.I	-
ApprovalVendorInvoiceListPMLayout.I	-
ApprovalVendorInvoicePMLayout.I	ApprovalVendorInvoicePMLayoutREST.I
ApprovalInvoiceDraftListLayout.I	-
ApprovalInvoiceDraftLayout.I	-
ApprovalInvoiceDraftLineLayout.I	ApprovalInvoiceDraftLineLayoutREST.I
ApprovalInvoiceDraftLinesLayout.I	ApprovalInvoiceDraftLinesLayoutREST.I
ApprovalExpenseSheetLinePMLayout.I	-
ApprovalMileageSheetLinePMLayout.I	-

Attention: The REST approval layouts are used only if the Maconomy system has long text enabled. For additional information, see the Appendix A: Screens, Layouts, and Metadata Files section in this document.

Search Layouts

MScript	REST
-	Find_AbsenceType.I
-	Find_Account.I
-	Find_Activity.I
-	Find_Employee.I
-	Find_JobFavorite_Timesheet.I **Note:** Starting Touch 3.6 API, The Find_JobFavorite_Timesheet.I is replaced by Find_JobFavorite.I. If you want to make changes specifically for timesheets, create a new layout file and add it to the LayoutList_REST.I. For example: ▪ Timesheet Filename: Find_JobFavorite_Timesheet.I Layoutname: #N"find_jobfavorite_timesheet" ▪ Expense Filename: Find_JobFavorite_Expensesheet.I Layoutname: #N"find_jobfavorite_expensesheet" ▪ Mileage Filename: Find_JobFavorite_Mileagesheet.I Layoutname: #N"find_jobfavorite_mileagesheet"
-	Find_JobHeader_Timesheet.I
-	Find_LocationFromTo.I

MScript	REST
-	Find_PurchaseOrderLine_ExpenseSheet.I
-	Find_PurchaseOrderLine_TimeSheet.I
-	Find_TaskListLine.I
-	FindJobHeader_Expense.I
-	FindJobHeader_PurchaseOrder.I
-	FindVendor_PurchaseOrder.I
	Find_FinanceVATCode.I

Purchase Order Layouts

MScript	REST
-	PurchaseOrderLayoutREST.I
-	PurchaseOrderLineLayoutREST.I
-	PurchaseOrdersLayoutREST.I
-	RejectionNotificationListLayoutREST.I

Absence and Allowance Requests Layouts

MScript	REST
-	AbsenceApprovalFormLayoutREST.I
-	AbsenceApprovalListLayoutREST.I
-	AbsenceRequestLayoutREST.I
-	AbsenceRequestsLayoutREST.I
-	AllowanceRequestLayoutREST.I
-	AllowanceRequestsLayoutREST.I

Attention: For additional information, see the Appendix A: Screens, Layouts, and Metadata Files section in this document.

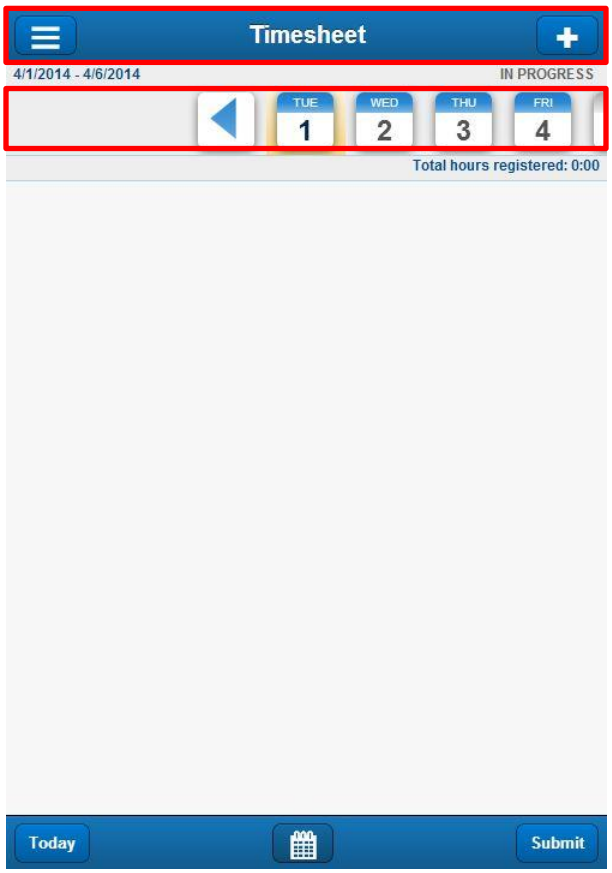
Layout Configuration

The layout customization has been introduced to help clients control fields that they want to display on and apply some styling or formatting to a Deltek Touch screen based on CSS technology.

Deltek Touch supports configuration of fields and columns on several screens (such as **Timesheet Line**, **Add Job**, **Approval**, and **Summary**), allowing you to show or hide fields that are available in the metadata. The layout customization capabilities, however, do not apply to all controls present on a particular screen. The common portion of the screen that you cannot customize is the title bar area, where you can usually find the screen name (for example, see red box with “Timesheet” in Figure 1).

Note: The layout customization capabilities do not apply to the **Weeks**, **Calendar** (both screen and carousel), **Settings**, and document viewer.

Figure 1: Timesheet Screen



For each Deltek Touch release, there is a set of standard layouts that controls screens in the application. When you install the Web service (using the FPU), these standard layouts are added to the **MaconomyMScript/Standard/Layouts** folder. The first time you customize a layout, you need to copy it to the Custom folder. Performing this allows you to preserve your customization when Deltek Touch is upgraded. If you change the standard layouts, all your changes are lost when upgrading Deltek Touch.

Note: If you upgrade Deltek Touch, you need to update manually your custom files (in the Custom folder), to reflect the potential changes in the standard files. In addition, you can only modify the **DeltekTouch.I** configuration file if you want to customize the Log Entry screen layout (Registrations) or the server settings.

Customization Capabilities

Deltek Touch now allows you to perform the following tasks:

- Add new fields based from Metadata files, which you can also find in the **Layouts** folder.
- Format a specific element on a screen
- Change the font color, size, or style (bold, italic, or underline)
- Add a margin and padding
- Apply an ellipsis for long text (truncation)
- Add a column to some screens (such as **Summary**).
- Interchange order of lines
- Remove, and modify fields from a layout.

Note: You need to log in again to Deltek Touch for the changes to take effect.

Approval

- You cannot customize the title and icons.
- You cannot customize action buttons (such as **Approve**, **Reject**, **Undo Approve/Reject**, and **Message to Employee**).
- You cannot customize columns with and .

Summary

- You cannot customize the upper portion of a screen, which includes the title, **Submit** button, and header details.
- You cannot customize the column with . You can only customize column headers and fields.

Log Entry

You can add any field from the table part of the SpeedSheet window (DailyTimeSheetLine database table), any field from the table part of the Time Sheets window (TimeSheetLine database table), and some of the variables from the table part of SpeedSheet window.

- Each field or variable displays on a new line.
- Variables are read-only.
- Fields can either be read-only or editable, depending on the data model from the server.
- You cannot customize fonts.

Map

You can allow users to use the Map feature, which opens the Google Map and displays the **From** and **To** route, on the **Expense Sheet Line** and **Mileage Sheet Line** screens.

The Map action is, by default, shown in the Mileage Sheet Line screen and hidden in the Expense Sheet Line screen.

Our recommendation is to use the Map functionality on Mileage Sheet Line, since this is a screen dedicated for registering mileage. In case you would like to register expense and mileage in the same screen, you would need to customize the Expense Sheet Line screen by adding the proper mileage fields (From, To, VehicleType…) and making the Map action visible, as iilustrated below:

```
actions: [
          { key: #N"newline", hidden: false },
          { key: #N"attach", hidden: false },
          { key: #N"map", hidden: true}
]
```

You can also apply this to the previous supported API versions (>=2.1), assuming that end users are using the Touch 3.2 (or newer) app.

Layout Components and Attributes

This section discusses the non-editable and editable layout components and corresponding attributes used in customization of Deltek Touch.

Non-Editable Layout Components

Use the non-editable layout components to display a non-editable text on a screen or non-Sencha or non-HTML5 component related. It generates HTML tags-based layouts.

Component	Description
Label	Basic component in the layout. It is responsible for displaying a caption or title on the screen.
Field	This component is used to display a field from the database (using the metadata file) on the screen.
LinearText	This component is used for grouping components, such as “Label” and “Field”. If you want to display a group of components in one line, you need to use this component.
Grid	This component is used for tabulating the data (in a columns and rows).
Column	This component used to define column in the grid component.

Editable Layouts Components

Use the editable layout components to display a server-variable controlled editable field. These components are only applicable to all Line screens, except for the **Add Job** and **Timesheet Line** screens.

Component	Description
TextField	This component is used to display a text field on a Line screen (see Figure 1).
DocumentField	This component is used to display a field on the Line screen intended for attachments. You need to define the “DocumentName” as the source field not the “DocumentArchiveNumber” field.
Form	This component is used mainly for “Line Screens” (for example, the **Log Entry** screen).
Group	This component creates a field set to group the “textfield” and “documentfield” components.

Component Layout Attributes

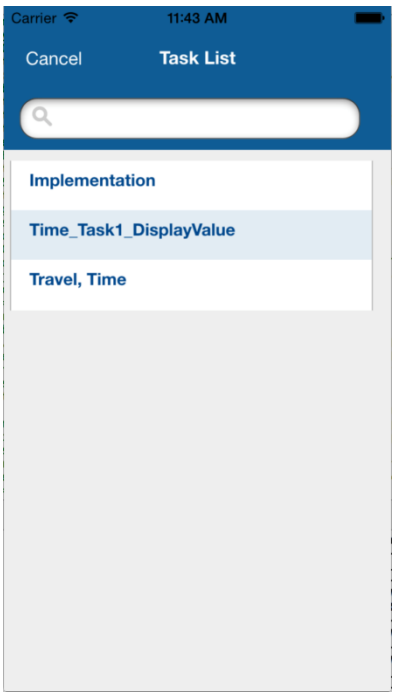
The following table describes component attributes, but not all of them can be used in a specific component:

Attribute	Description
type	This is used to specify which component to use in a layout. All of the layout files already contain the component that you can use in customizing a screen. This is one of the most commonly used attributes in the layouts.
region	This is used to state the name of the portion of a screen.
protectedregion	This is used to identify whether the portion of the screen in the Touch client is customizable or not.
style	This is used to describe the formatting or styling that needs to be applied in a specific component.

cls	The cls (short term for class) attribute is used to format the layout of pages. It defines text styles, columns, and other aspects of pages, providing an alternative means of defining the style of each block of text within a page. Once the style is defined in cls, it can be used by any page that references the CustomLayout.scss file.
	Sample Code:
	`{`
	`type: #N"lineartext",`
	`cls:#N"text_medium_bold"`
	`}`

purpose	You can use this to state the whole description of a certain component or region in the layout file.
detail	This is only used for “grid” component. This attribute represents the content of grid. It refers to “row” in the grid.
header	This is only used for “grid” component. This attribute is used for describing the header title in the grid; it refers to the “column title” in the grid. Not all “grid” have a header component. The layout file already contains a header if the grid can have a “detail” or a “header.” This is dependent on the Sencha component used in the Touch client.
items	This is only used for the “lineartext” and “column” components. It is a container for the other component.
mask	This is only used in the “field” and “textfield” component. The purpose of this attribute is the following: ▪ If the field value needs to use the device short-date formatting.

Attribute	Description
	▪ If the field value is not informative but predictable (for example, Status) and you want to set a descriptive value when it displays on a screen. ▪ If the field value needs to use the Touch Standard hours formatting (for example, 2:30). ▪ If the requirements need to display the amount and currency value (for example, DKK 200.00). ▪ If the field needs to show the “Untitled” string, instead of empty string.

lookuplayoutname	This is only used for customizing the layout and search field of the lookup. You can customize the following components in the lookup:
	▪ **Search Field Area** — This area is composed of the text field component with a magnifying glass icon (see screenshot below). By entering characters in this component, it filters the records in the list.
	▪ **Lookup List Area** — This area is composed of the list component used for displaying the records that are retrieved from the Web service (see the list in the screenshot below).
	Using this attribute in the lookup field component is pointing to another layout file. You need to add this to the DeltekTouchWS.ms file.
	 
	
	Sample Code
	Adding the “lookuplayoutname” to the lookup field component:
	`{`
	`source: #T"Deliverable",`

Attribute	Description

	`label: #N"Remark",`
	`searchlayout: #N"ActivitySearch",`
	`type: #N"lookupfield",`
	`queryfieldname: #N"KeyValue",`
	`displayfields: [`
	`{`
	`source: #N"ActivityTextVar",`
	`queryfieldname: #N"DisplayValue"`
	`},`
	`{`
	`label: #N" - ",`
	`},`
	`{`
	`source: #N"ActivityNumber",`
	`queryfieldname: #N"DisplayValue"`
	`}`
	`]`
	`Lookuplayoutname: "DeliverableLookupLayout"`
	`}`
	
	Adding the layout to the DeltekTouchWS.ms file:
	`If (inpJSON.operation == "getlayouts")  {`
	`  Sessions var Debug;`
	`  localDebug = false;`
	`  outputJSON.sessionid = currentsession();`
	`  outputJSON.operation = "getlayouts";`
	`  outputJSON.ok = true;`
	`  new outputJSON.CustomLayout = {};`
	`  outputJSON.CustomLayout = [];`
	`  new outputJSON.CustomLayout [0] = DeltekTouchLayout::getlayout ("TimesheetLayout.I","timesheetLayout");`
	`  new outputJSON.CustomLayout [1] = DeltekTouchLayout::getlayout ("SummaryLayout.I","summaryLayout");`
	`  new outputJSON.CustomLayout [51] = DeltekTouchLayout::getlayout ("DeliverableLookupLayout.I","deliverableLayout");`

Attribute	Description

	Content of a Lookup Layout File:
	`{`
	`Layoutname: #N”DeliverableLookupLayout”,`
	`Layout: [`
	`{ region: #”ToolbarArea”,`
	`protectedregion: true`
	`},`
	`{`
	`region: #N”SearchFieldArea”,`
	`protectedregion; false,`
	`type: #N”serachfield”`
	`filterkeys: “KeyValue, DisplayValue”,`
	`placeholder: “Task Name, Task Description”`
	`},`
	`{`
	`region: #N”LookupListArea”,`
	`protectedregion: false,`
	`type: #N”grid”,`
	`detail: [`
	`{`
	`type: #N”column”,`
	`style: { width: #N”75%” },`
	`items: [`
	`{`
	`type: #N”lineartext”,`
	`style: {`
	`fontsize: #N”14px”,`
	`color: #N”00488a”,`
	`fontweight: #N”bold”`
	`},`
	`items: [`
	`{`
	`type: #N”field”,`
	`source:`
	`#N”KeyValue”`
	The content of the lookup layout file contains three regions::
	▪ **ToolbarArea** — This is a protected region.

Attribute	Description

	▪ **SearchFieldArea** – This represents the text field component in the lookup.
	▪ **LookupListArea** — This represents the list component of the lookup.
	Adding this attribute to the “lookupfield” component only works if the Web method used by the lookup to retrieve the data is the “search” operation or the lookup uses the search layout from DeltekTouch.I.
	New layout attribute syntax:
	`options : {`
	`restriction: #N"samplequery = 1"`
	`}`

restriction	This is used by the lookup component, specifically for REST enabled system. The purpose of this attribute is to dynamically filter the data coming from the response.
	Sample Code:
	`{`
	`label: #T"SelectedOption2 - County",`
	`source: #N"selectedoption2",`
	`type: #N"lookupfield",`
	`queryfieldname: #N"name",`
	`options: {`
	`restriction: #N"remark1 = selectedoption1"`
	`},`
	`displayfields: [`
	`{`
	`source: #N"selectedoption2",`
	`queryfieldname: #N"name"`
	`}`
	`]`
	`}`

Mandatory	The purpose of this attribute is to force the user to put a value in a certain field. Rest enabled system can use this attribute.
	Sample Code:
	`{`
	`source: #N"NumberOf",`
	`label: #T"Quantity",`
	`mandatory: #N”true’`

Attribute	Description
	} This does not applied to readonly and disabled fields.

Style Layout Attributes

The following table contains the style and formatting attributes in a layout. All of the styling attributes that you can define in the layout are derived from “CSS” technology, including how the styles are applied to a certain component.

Attribute	Description

fontsize	This is used to define the size of a font. It accepts either in “px” (by pixel) or “em” measurement unit.
	Example:
	`style: {`
	`fontsize: “12px”`
	`}`
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/pr_font_font-size.asp.

textdecoration	This is usually used to add an underline/overline in the text.
	Example:
	`style: {`
	`textdecoration: “underline”`
	`}`
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/pr_text_text-decoration.asp.

textalignment	This is used to define the alignment of a text. You can set one of the follwing common values in this attribute:
	▪ "right"
	▪ “left”
	▪ “center”
	Example:
	`style: {`
	`textalignment: “left”`
	`}`
	Attention: For more details about the value you can define, see

Attribute	Description
	http://www.w3schools.com/cssref/pr_text_text-align.asp.

fontstyle	This is used to define the style you want to apply to a text. The common value you can set in this attribute is “italic.”
	Example:
	`style: {`
	`fontstyle: “italic”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_font_font-style.asp.

fontweight	This is used to define how thick or thin the text will be displayed on a screen. The common value you can set is “bold.”
	Example:
	`style: {`
	`fontweight: “bold”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_font_weight.asp.

margintop	This is used to define the top margin of a component on a screen. This attribute usually accepts value in “px” (by pixel) measurement unit.
	Example:
	`style: {`
	`margintop: “12px”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_margin-top.asp.

marginbottom	This is used to define the bottom margin of a component on a screen. This attribute usually accepts value in “px” (by pixel) measurement unit.
	Example:
	`style: {`
	`marginbottom: “12px”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_margin-bottom.asp.

Attribute	Description

marginleft	This is used to define the left margin of a component on a screen. This attribute usually accepts value in “px” (by pixel) measurement unit.
	Example:
	`style: {`
	`marginleft: “12px”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_margin-left.asp.

marginright	This is used to define the right margin of a component on a screen. This attribute usually accepts value in “px” (by pixel) measurement unit.
	Example:
	`style: {`
	`marginright: “12px”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_margin-right.asp.

textoverflow	This is used to define the result if the text overflows in a container. The common scenario is that, if the field value you are expecting is too long and does not fit into the screen or column width, you can truncate the text by specifying an “ellipsis” (for example, This text is too long...).
	Example:
	`style: {`
	`textoverflow: “ellipsis”`
	`}`
	Sometimes, however, if you define an ellipsis styling, it does not apply automatically to the screen. The usual reason is that, there is some external styling attribute that hinders it. To apply the styling successfully, you can add another styling attribute, which is the “overflow.” The most common value you can set on this is “hidden.”
	Example:
	`style: {`
	`textoverflow: “ellipsis”,`
	`overflow: “hidden”`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/css3_pr_text-overflow.asp.

Attribute	Description

paddingtop	This is used to create padding at the top side of a component. This attribute usually accepts value in “px” (by pixel) measurement unit and does not accept any negative value.
	Example:
	`style: {`
	`paddingtop: “16px”,`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_padding-top.asp.

paddingleft	This is used to create padding on the left side of a component. This attribute usually accepts value in “px” (by pixel) measurement unit and does not accept any negative value.
	Example:
	`style: {`
	`paddingleft: “16px”,`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_padding-left.asp.

paddingright	This is used to create padding on the right side of a component. This attribute usually accepts value in “px” (by pixel) measurement unit and does not accept any negative value.
	Example:
	`style: {`
	`paddingright: “16px”,`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_padding-right.asp.

paddingbotom	This is used to create padding at the bottom side of a component. This attribute usually accepts value in “px” (by pixel) measurement unit and does not accept any negative value.
	Example:
	`style: {`
	`paddingbottom: “16px”,`
	`}`
	Attention: For more details about the value you can define, see

Attribute	Description
	http://www.w3schools.com/cssref/pr_padding-bottom.asp.

whitespace	This is used to control how the white space inside the element will be handled.
	Example:
	`style: {`
	`whitespace: "nowrap"`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_text_white-space.asp.

backgroundcolor	This is used for applying a background color of an element.
	Example:
	`style: {`
	`backgroundcolor: "#00FF00"`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_background-color.asp.

backgroundimage	This is use for applying background images of an element.
	Example:
	`style: {`
	`backgroundimage: #N"-webkit-linear-gradient(top, #ff1a1a,#e60000 3%,#b30000)"`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/pr_background-image.asp.

borderradius	This is used for applying border radius of an element.
	Example:
	`style: {`
	`borderradius: #N"2em"`
	`}`
	Attention: For more details about the value you can define, see
	http://www.w3schools.com/cssref/css3_pr_border-radius.asp.

bordercolor	This is used for applying border color of an element. Example:

Attribute	Description

	`style: {`
	`bordercolor: #N"#ff0000 #0000ff"`
	`}`
	The border color will apply in the top and right border of an element.
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/pr_border-color.asp.

webkitbackgroundclip	This is used to specify the painting area of the background.
	Example:
	`style: {`
	`webkitbackgroundclip: #N"padding-box"`
	`}`
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/css3_pr_background-clip.asp.

minwidth	This is used to specify the minimum width of an element.
	Example:
	`style: {`
	`minwidth: #N"10px"`
	`}`
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/pr_dim_min-width.asp.

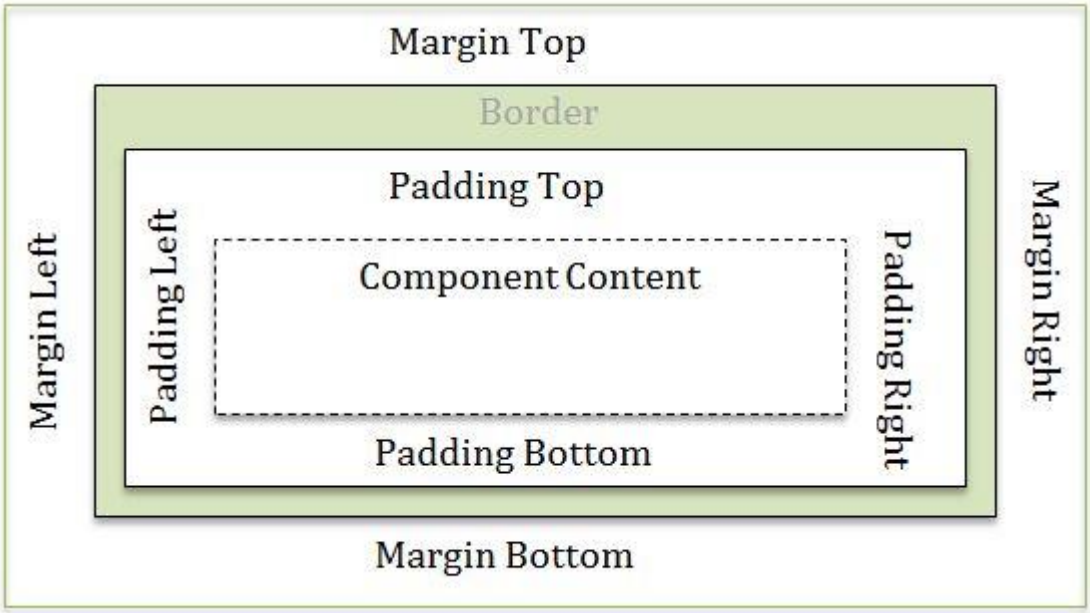
maxwidth	This is used to specify the maximum width of an element.
	Example:
	`style: {`
	`maxwidth: #N"10px"`
	`}`
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/pr_dim_max-width.asp.

action	This is used to set the action for the link label.
	
	Example:
	`type: #N”lineartext”,`
	`cls: #N”link_label_margintop30”,`
	`action: #N”open_expensesheet_header”,`

Attribute	Description

	`items: {`
	`title: #T"Expense Sheet",`
	`type: #N"linklabel"`
	`}`
	Attention: For more details about the value you can define, see http://www.w3schools.com/cssref/pr_dim_max-width.asp.

Margin and Padding Box Model



Class Layout Attributes

The following table contains the style and formatting attributes used in a Custom layout.

Attribute	Description

CssClass	Object that is used to classify a custom style. It is placed in the CssClass.i file and defined by Maconomy to minify style from the layout. In addition, it is helpful in formatting other aspects of the page layout.
	This new attribute of Maconomy gives you an option to overrule how a field looks.
	Example:
	`{CssClass:[`
	`{`

Attribute	Description

	`className: #N"text_medium_bold",`
	`style: {`
	`fontsize: #N"16px!important",`
	`textalignment: #N"left",`
	`color: #N"#f2b600"`
	`}`
	}
	]}

className	Under CssClass object is the ClassName, which is used to define the cls that the user wants to update.
	Example:
	`{`
	`className: #N"text_medium_bold",`
	`style: {`
	`fontsize: #N"16px!important",`
	`textalignment: #N"left",`
	`color: #N"#f2b600"`
	`}`
	}

CustomLayout.scss

To improve the way Deltek Touch uses styles in layouts, you can use the CustomLayout.scss CSS style. Using the customLayout file minifies the layout file size from the standard folder.

Field Types

The following table displays the CSS standard layout that you can customize:

LinearText	Column	Fields

`text_lookup{`
`color: #00488a;`
`}`
`.text_center{`
`text-align: center;`
`}`
`.text_right{`
`text-align:right;`
`}`
	`.columntext_15width_right{`
	`margin-left: 15px!important;`
	`text-align:right;`
	`}`
	`.columntext_right_30right{`
	`text-align:right;`
	`padding-right: 30px;`
	`}`
		`.field_border_radius{`
		`font-size:16px;`
		`height: 30px;`
		`width:30px;`
		`background-color:#cc3300;`
		`border-radius: 50%;`
		`display: inline-block;`
		`color: #ffffff;`

LinearText	Column	Fields

.text_left{
text-align:left;
}
.text_upper{
text-transform:uppercase;
}
.text_bold{
font-weight: bold;
}
.text_marker{
color: #f2b600;
}
.text_marker_bold{
color: #f2b600;
font-weight: bold;
}
.text_required{
color: #C71B05;
}
.text_ellipsis {
white-space:nowrap;
text-overflow:ellipsis;
overflow: hidden;
}
.text_xxsmall_comment{
font-size: 9pt;
color: #f2b600;
padding: 10px 10px 0px;
}
.text_xxsmall_comment_5left{
font-size: 9pt;
color: #f2b600;
padding-left: 5px;
}
.text_xxsmall_nowrap{
font-size: 9pt;
white-space: nowrap;
}
.text_xxsmall_right{
font-size: 9pt;
text-align: right;
}
	.columntext_left_20left{
	text-align:left;
	padding-left: 20px;
	}
	.columntext_20left{
	padding-left: 20px;
	}
	.columntext_right{
	text-align:right;
	}
	.columntext_center{
	text-align:center;
	}
	.columntext_center_20perwidth{
	text-align:center;
	width: 20%;
	}
	.columntext_small{
	font-size: 12px;
	}
	.columntext_small_color{
	font-size: 12px;
	color: #475262;
	}
	.columntext_small_10bottom{
	font-size: 12px;
	margin-bottom: 10px;
	}
	.columntext_small_right{
	font-size: 12px;
	text-align:right;
	}
	.columntext_small_padding{
	font-size: 12px;
	padding-left: 35px;
	}
	.columntext_small_35perwidth{
	font-size: 12px;
	width: 35%;
	}
	.columntext_small_35perwidth_right{
	font-size: 12px;
		line-height: 28px;
		text-align: center;
		}
		.field_medium{
		font-size:14px;
		}
		.field_large_bold_30height{
		font-size: 16px;
		font-weight: bold;
		color: #434E5E;
		line-height: 30px;
		}
		.margin_bottom10{
		margin-bottom: 10px;
		}

LinearText	Column	Fields

.text_xxsmall_nowrap_8left
{
font-size: 9pt;
color: #475262;
white-space: nowrap;
padding-left: 8px;
}
.text_xxsmall_bold_right{
font-size: 9pt;
color: #475262;
font-weight: bold;
text-align: right;
}
.text_xsmall{
font-size: 10px;
}
.text_xsmall_status{
font-size: 10px;
color: #a00;
}
.text_xsmall_marker{
font-size: 10px;
color: #f2b600;
}
.text_small{
font-size: 12px;
}
.text_small_lookup{
font-size: 12px;
color: #00488a;
}
.text_small_note{
font-size: 12px;
color: #6b6b6b;
}
.text_small_note_50left{
font-size: 12px;
color: #6b6b6b;
padding-left: 50px;
}
.text_small_wrap{
font-size: 12px;
color: #475262;
}
.text_small_nowrap{
font-size: 12px;
	width: 35%;
	text-align:right;
	}
	.columntext_small_20perwidth{
	font-size: 12px;
	width: 20%;
	}
	.columntext_small_20perwidth_right{
	font-size: 12px;
	width: 20%;
	text-align:right;
	}
	.columntext_small_center{
	font-size: 12px;
	text-align:center;
	color: #434E5E;
	}
	.columntext_small_dark{
	font-size: 12px;
	color: #434E5E;
	}
	.columntext_small_center_15perwidth{
	font-size: 12px;
	text-align:center;
	color: #434E5E;
	width: 15%;
	}
	.columntext_small_center_25perwidth{
	font-size: 12px;
	text-align:center;
	color: #434E5E;
	width: 25%;
	}
	.columntext_small_center_25perwidth{
	font-size: 12px;
	text-align:center;
	color: #434E5E;
	width: 25%;
	}	

LinearText	Column	Fields

color: #475262;
white-space: nowrap;
}
.text_small_8right_right{
font-size: 12px;
color: #475262;
padding-right: 8px;
text-align:right;
}
.text_medium{
font-size: 14px;
color: #475262;
}
.text_medium_right{
font-size: 14px;
color: #475262;
text-align:right;
}
.text_medium_bold{
font-size: 14px;
font-weight: bold;
color: #475262;
}
.text_medium_bold_right{
font-size: 14px;
font-weight: bold;
color: #475262;
text-align:right;
}
.text_medium_bold_lookup{
font-size: 14px;
font-weight: bold;
color: #00488a;
}
.text_medium_bold_35perwid
h{
font-size: 14px;
font-weight: bold;
color: #475262;
width: 35%;
}
.text_medium_bold_required
{
color: #C71B05;
font-size: 14px;
font-weight: bold;		

LinearText	Column	Fields

}
.text_medium_bold_dark{
font-size: 14px;
font-weight: bold;
color: #434E5E;
}
.text_medium_dark_right{
font-size: 14px;
color: #434E5E;
text-align:right;
}
.text_medium_bold_dark_right{
font-size: 14px;
font-weight: bold;
color: #434E5E;
text-align:right;
}
.text_medium_bold_dark_15right{
font-size: 14px;
font-weight: bold;
color: #434E5E;
margin-right: -15px!important;
}
.text_large_bold{
font-size: 16px;
font-weight: bold;
color: #475262;
}		

Rules

- You can only override style based from CustomLayout.scss.
- When defining a style, always declare the "className" attribute first before the style attribute.
- After adding the CssClass.I file, make sure the file is in the first line in the LayoutList_MScript.I file.

> **Attention:** For more information, see the Change the Style subsection in this document.

Sample Code

Below is a sample code in changing the color:

```
{CssClass:[
        {
```

```
                className: #N"text_medium_bold",
                style: {
                        fontsize: #N"16px!important",
                        textalignment: #N"left",
                        color: #N"#f2b600"
                }
        }
]}
```

Old and New Styling Methods

The purpose of this section is to provide a comparison between the two styling methods using sample codes.

Old Styling Method ("Style" Layout Attribute):

ApprovalExpenseSheetListLayout.l

Touch 3.2 - Maconomy < 2.4 old styling method

```
1 {
2   layoutname: #N"ApprovalExpenseSheetListLayout",
3   layout:
4   [
5     {
6       protectedregion: true,
7       region: #N"TitlebarArea"
8     },
9     {
10      protectedregion: false,
11      region: #N"ExpenseSheetList",
12      type: #N"grid",
13      detail: [
14        {
15          type: #N"column",
16          style: {
17            fontsize: #N"12px"
18          },
19          items: [
20            {
21              type: #N"lineartext",
22              style: {
23                fontsize: #N"14px",
24                fontweight: #N"bold",
25                color: #N"#434E5E"
26              },
27              items: [
28                {
29                  type: #N"field",
30                  source: #N"EmployeeName"
31                },
32                {
33                  type: #N"label",
34                  title: #N" - "
35                },
36                {
37                  type: #N"field",
38                  source: #N"EmployeeNumber"
39                }
40              ]
41            },
42            {
43              type: #N"lineartext",
44              style: {
45                fontsize: #N"14px",
46                fontweight: #N"bold",
47                color: #N"#434E5E"
48              },
49              items: [
50                {
```

Note: You can only use the old styling method in all Touch layouts if you are on a Maconomy version lower than 2.4.4.

New Styling Method ("cls" Layout Attribute)

DeltekTouchWS.ms | DeltekTouch.l | VendorInvoiceJournal | ApprovalExpenseSheet

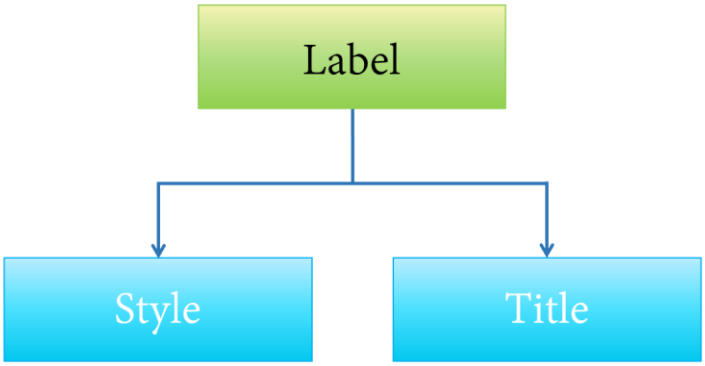
Touch 3.2 - Maconomy 2.4 new styling method

```
1 {
2   layoutname: #N"ApprovalExpenseSheetListLayout",
3   layout:
4   [
5     {
6       protectedregion: true,
7       region: #N"TitlebarArea"
8     },
9     {
10      protectedregion: false,
11      region: #N"ExpenseSheetList",
12      type: #N"grid",
13      detail: [
14        {
15          type: #N"column",
16          cls:#N"columntext_small",
17          items: [
18            {
19              type: #N"lineartext",
20              cls:#N"text_medium_bold_dark",
21              items: [
22                {
23                  type: #N"field",
24                  source: #N"EmployeeName"
25                },
26                {
27                  type: #N"label",
28                  title: #N" - "
29                },
30                {
31                  type: #N"field",
32                  source: #N"EmployeeNumber"
33                }
34              ]
35            },
36            {
37              type: #N"lineartext",
38              cls:#N"text_medium_bold_dark",
39              items: [
40                {
41                  type: #N"field",
42                  source: #N"ExpenseSheetNumber"
43                }
44              ]
45            },
46            {
47              type: #N"lineartext",
48              items: [
49                {
50                  type: #N"field",
```

Note: You can only use the new styling method in all Touch layouts if you are on Maconomy version 2.4.4 or higher.

Syntax Descriptions

Label



Description

Label is one of the basic components in the layout customization. It is responsible for displaying literal string or caption in the screen. It is commonly used for defining a header title in the “grid” component or often time used for description of the “field” component.

Rules

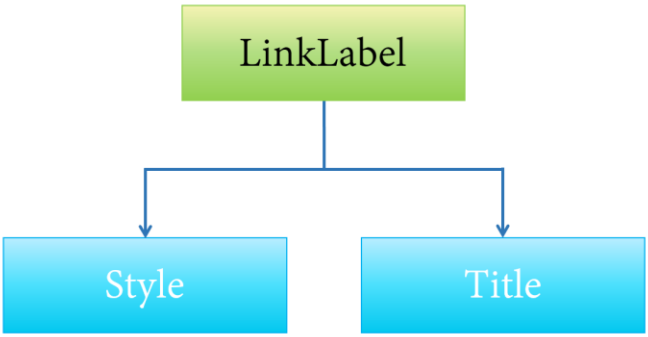
- When defining a label always declare the “title” attribute.
- The “style” attribute is optional.
- It is commonly used for defining a header title in the “grid” component or often time used for description of the “field” component.

Example

```
{
        type: “label”,
        title: #T “Employee Number: “
},
{
        type: “field”,
        source: “EmployeeNumber”
}
```

- Output: Employee Number:001

LinkLabel



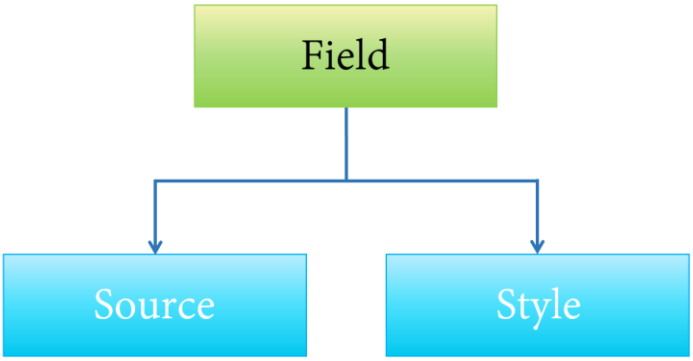
Description

This is just the same with the label but it has an added attribute of adding action to the label. This is clickable and can be assigned an action.

Example

```
{
        type: "linklabel",
        title: #T "Expense sheet",
        action: #N "open_expensesheet_header"
},
```

Field



Description

Use to display a database field on a screen based from a Metadata file that can be found in the Standard/Layouts folder.

```
{
      {region: #K"TotalHoursRegistered",
       operation: #K"getperiod",
       metadata:
                [
                                    // Maconomy Web Service Variables
                                    // targetnode: #N"{DayTotals}"
                  {source: #K"TheDate"}, // the date in long format
                  {source: #K"ShortDate"}, // the date in short format
(mm/dd)
                  {source: #K"WeekDay"}, // the name of the week day
(Monday...)
                  {source: #K"ExternalRegisteredHours"}, // external hours
registered on the given day/date, on all time sheet lines
                  {source: #K"InternalRegisteredHours"}, // internal hours
registered on the given day/date, on all time sheet lines
                  {source: #K"TotalRegisteredHours"}, // total hours
registered on the given day/date, on all time sheet lines
                  {source: #K"FixedHours"}, // fixed hours on the given
day/date (how many hours do you expect to work this day)
                  {source: #K"OvertimeHours"} // overtime hours on the given
day/date

              ]
}
```

Rules

- The “source” attribute is very important when declaring a “field” component.
- The “style” attribute is optional.
- The “mask” attribute can be used only for the “field” component (see the Component Layout Attributes). As described in the Component Layout Attributes section, the mask can be used
- If the value of the field does not give descriptive information on a screen
- The value of the field is very predictable (for example, the Timesheet Status can have the following values: A = Approved, R = Rejected, or S = Submitted).
- If the value of the field can be empty string and you would like to show “Untitled” instead of empty string

Examples

This is a simple way to declare a “field” component in the layout file. It displays the “EmployeeNumber” field with “Employee Number: “ label.

```
{
  type: “label”,
  title: #T “Employee Number: “
```

```
},
{
   type: "field",
   source: "EmployeeNumber"
}
```

Output: Employee Number:001

If you want to set a specific styling in the "field" component, you can use the "style" attribute. For example, you want to display "EmployeeNumber" in blue.

```
{
   type: "label",
   title: #T "Employee Number: "
},
{
   type: "field",
   source: "EmployeeNumber",
   style: {
         color: "#00488a"
   }
}
```

Output: Employee Number:001

> **Note:** The value of color in this example is the corresponding hexadecimal value of color blue in CSS. If you want to have another styling, refer to the Style Attributes section.

Another case of the "field" component is that, when the field value is not bringing descriptive information on the screen and when you want it that field value be informative, you can use the "mask" attribute. For example, the "field" can only contain "S," "A," or "R" value, and you want to display it as "Submitted," "Approved," and "Rejected," respectively.

```
{
   type: "label",
   title: #T "Timesheet Status "
},
{
   type: "field",
   source: "Status",
   mask: {
         Description: {
```

```
                S: #T"Submitted",
                A: #T"Approved",
                R: #T"Rejected"
            }
    }
}
```

Output

- If the “Status” field value is “S”, the output is:

Timesheet Status: Submitted

- If the “Status” field value is “A”, the output is:

Timesheet Status: Approved

- If the “Status” field value is “R”, the output is:

Timesheet Status: Rejected

> **Note:** In the example above, the “Status” field is very predictable because it can only contained “S,” “A,” or “R” value. Appending “#T” to the corresponding string indicates that the language translation is automatic.
>
> You cannot specify any styling attribute on the mask attribute. In case, there is a corresponding color for each of this value, you can use the “Color” attribute of the mask (see the next example).

To continue the example of the “field” component with the “mask” attribute, we are apply the corresponding color for each value of the “Status” field, such as green, blue and red for “S,” “A,” and “R,” respectively.

```
{
        type: "label",
        title: #T "Timesheet Status "
},
{
        type: "field",
        source: "Status",
        mask: {
            Description: {
                S: #T"Submitted",
                A: #T"Approved",
                R: #T"Rejected"
            },
            Color: {
```

```
                    S: "#53a800",
                    A: "#0080ff",
                    R: "#a00"
            }
        }
}
```

Output

- If the "Status" field value is "S", the output is:

Timesheet Status: Submitted

- If the "Status" field value is "A", the output is:

Timesheet Status: Approved

- If the "Status" field value is "R", the output is:

Timesheet Status: Rejected

Another scenario is when the value of the field can be a "true" or "false," but instead of defining the value of that field as an attribute within the "Description" and/or "Color" attribute, you need to set the "truevalue" or "falsevalue". If the value is set directly, an error in the layout file occurs. For example, we have the "Invoiceable" field, which contains "true" or "false" value, but we want to display a "Billable" or "Non-Billable" string on a screen.

```
{
        type: "field",
        source: "Invoiceable",
        mask: {
                Description: {
                        truevalue: #T"Billable",
                        falsevalue: #T"Non-Billable"
                }
        }
}
```

Output

- If the "Invoiceable" field value is "true", the output is:

Billable

- If the "Invoiceable" field value is "false", the output is:

Non-Billable

You can also add the corresponding color for each value of that field (see example below).

```
{
        type: "field",
```

```
        source: "Invoiceable",
        mask: {
                Description: {
                        truevalue: #T"Billable",
                        falsevalue: #T"Non-Billable"
                },
                Color: {
                        truevalue: "blue",
                        falsevalue: "red"
                }
        }
}
```

Output

- If the "Invoiceable" field value is "true", the output is:

Billable

- If the "Invoiceable" field value is "false", the output is:

Non-Billable

Some values of the field require specific formatting requirement or need to follow the device formatting, for example, the hours and date fields.

- Most of the time, the "hours" field value is in the "numeric/decimal" format (for example, 1.5 means 1 hour and 30 minutes). In this way, you need to declare the "hours" value in the "mask" attribute (see example below).

```
{
        type: "field",
        source: "NumberOf",
        mask: "hours"
}
```

- If you have a "date" field and the date format is in the "YYYY.mm.dd" form (for example, 2014.04.02) and you want to convert the format to the device short date formatting, you can set the "shortdate" value in the "mask" attribute (see example below).

```
{
        type: "field",
        source: "EntryDate",
        mask: "shortdate"
}
```

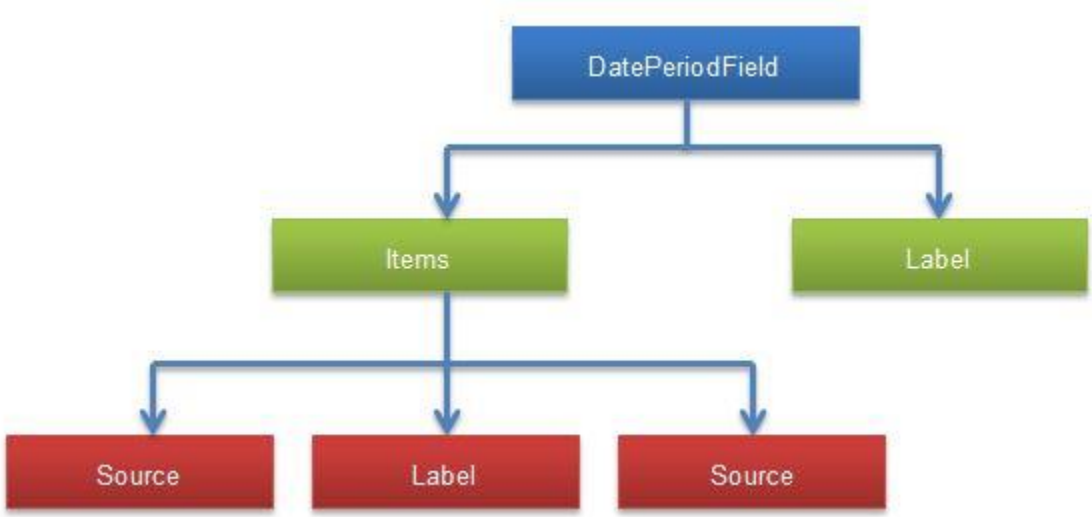
Note: YYYY represents the year (for example, 2014). **dd** represents the day of the month (from 00 - 31). **mm** represents the month of the year (from 01 – 12).

If you would like to show the “Untitled” string instead of an empty string when the value of a field is empty string, then you can set the “untitledtext” value in the “mask” attribute (see example below):

```
{
        type: #N"field",
        source: #N"description",
        mask: #N"untitledtext"
}
```

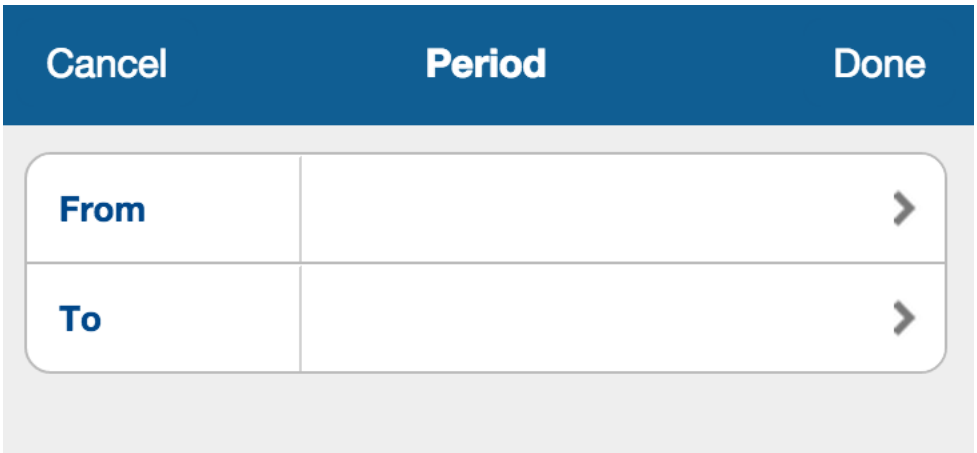
This is used, for example in the Expense Sheets screen, if the expense sheet does not have a description.

Date Period Field



Description

This component is used to enter and display a date period.



Rules

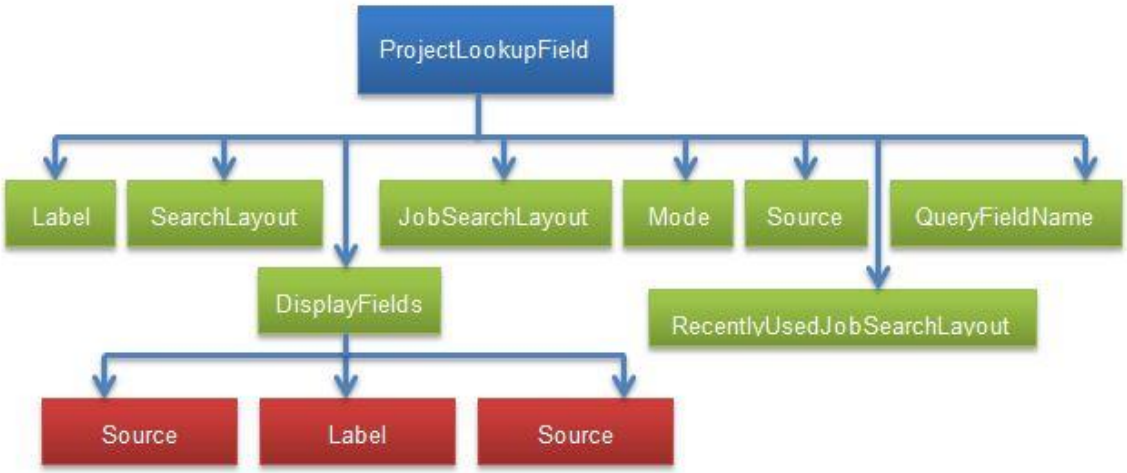
- This component does not support any styling.
- Items always contain a source, label, and another source in exact order.
- The source must be of type **Date**.
- The first label corresponds to label of the field. The second label act as a separator for the two dates.

Examples

This is a general syntax declaration of the “dateperiodfield” component:

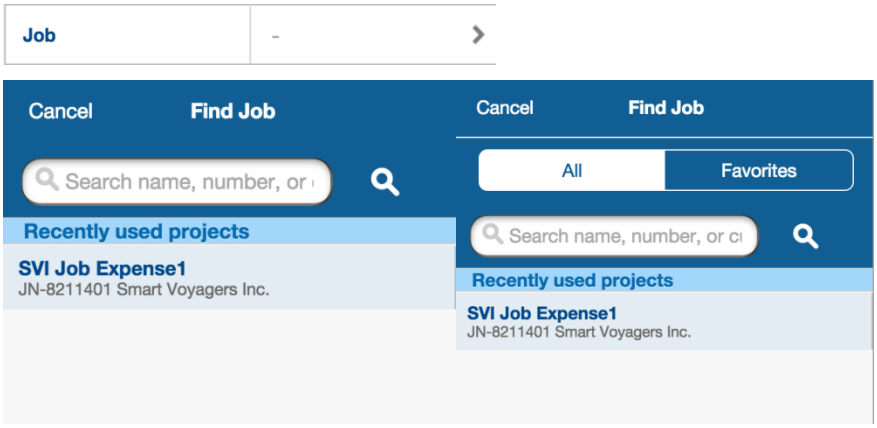
```
{
        type: “dateperiodfield”,
        label: “Period”,
        items: [
                { source: “FromDate” },
                { label: “-“ },
                { source: “ToDate” }
        ]
}
```

Project Lookup Field



Description

This component is used for entering a job using a lookup, it could also be used to enter a job based on favorite.



Rules

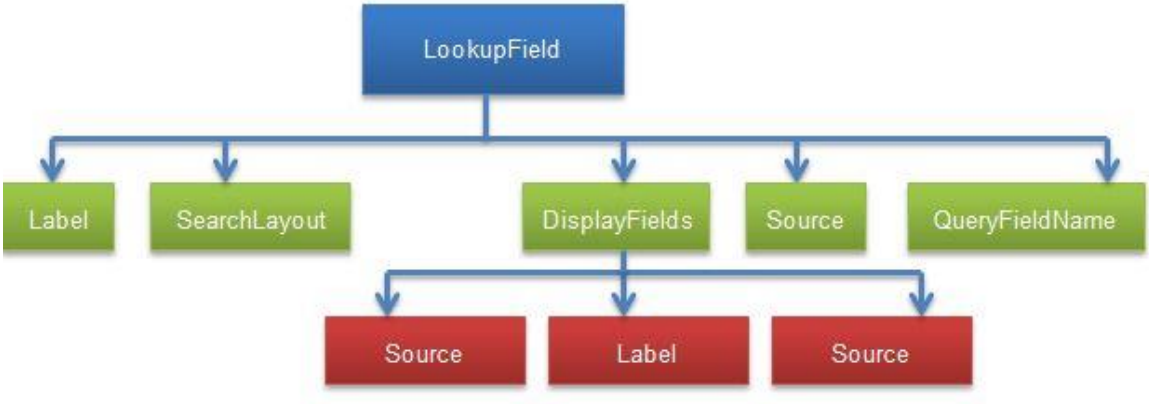
- This component does not support any styling.
- This component have three modes: “project”, “favorite,” and “projectfavorite.” This would determine how the job selection would be.
- This component should have a “queryfieldname = KeyValue” attribute in the parent node.
- The source attribute inside “displayfields” should contain a corresponding “queryfieldname.”
- The “queryfieldnames” can be a substitute attribute name for “queryfieldname.” This is needed if the lookup feature is composed of “All” and “Recently Used” because there are instance that these two features are requested with two different Web service methods, which returns different field names.
- Example: queryfieldnames: [“KeyValue1”,”KeyValue2”]

Examples

This is a general syntax declaration of the “projectlookupfield” component:

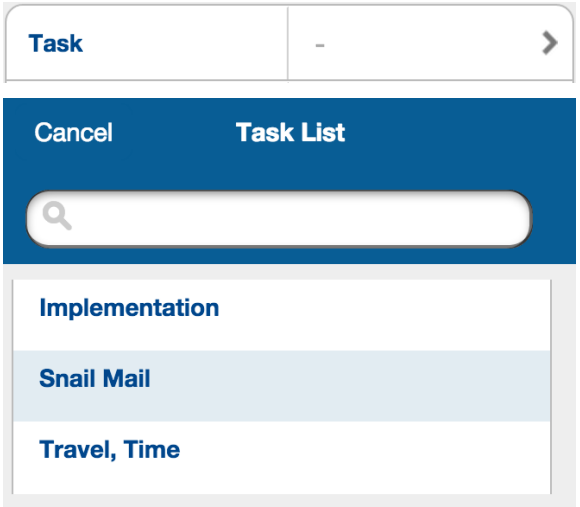
```
{
        source: “JobNumber”,
        label: “Job”,
        searchlayout: “Expense_JobSearch”,
        type: “projectlookupfield”,
        mode: "projectfavorite”,
        jobsearchlayout: “Expense_JobSearch”,
        recentlyusedjobsearchlayout: “Expense_RecentlyUsedJobSearch”
        queryfieldname: “KeyValue”,
        displayfields: [
                {
                        source: “JobNumber”,
                        queryfieldname: “KeyValue”
                },
                {
                        label: “ – “,
                },
                {
                        source: “JobNameVar”,
                        queryfieldname: “DisplayValue”
                }
        ]
}
```

Lookup Field



Description

This component is used for entering a value using a lookup, which is commonly used in entering tasks and activities.



Rules

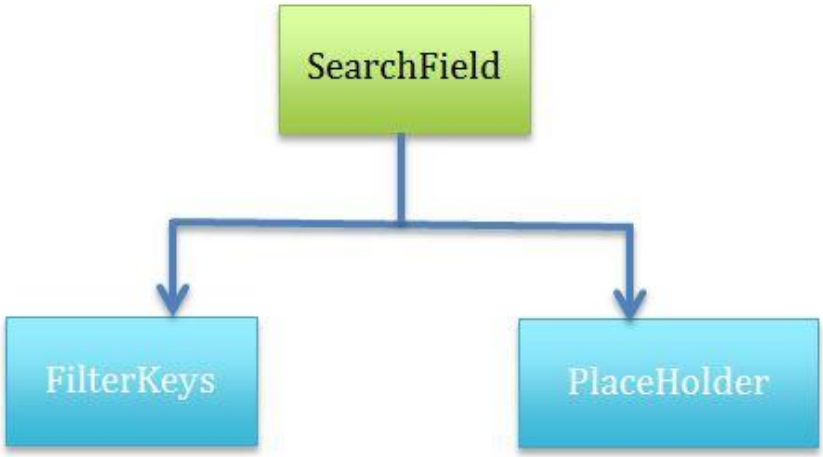
- This component does not support any styling.
- This component should have an attribute called “searchlayout.”
- The “source” attribute inside displayfields should have a corresponding “queryfieldname.”
- The “queryfieldnames” can be a substitute attribute name for “queryfieldname.” This is needed if the lookup feature is composed of “All” and “Recently Used” because there are instance that these two features are requested with two different web service methods, which returns different field names.
- Example: queryfieldnames: [“KeyValue1”,”KeyValue2”]

Examples

This is a general syntax declaration of the “lookupfield” component:

```
{
        source: “TaskName”,
        label: “Task”,
        searchlayout: “Expense_TaskSearch”,
        type: “lookupfield”,
        queryfieldname: “KeyValue”,
        displayfields: [
                {
                        source: “TaskName”,
                        queryfieldname: “KeyValue”
                },
                {
                        label: “ – “,
                },
                {
                        source: “TaskDescriptionVar”,
                        queryfieldname: “DisplayValue”
                }
        ]
}
```

SearchField



Description

This component is only used for the lookup layout file, which is being referenced from the “lookupfield” component. To control what field will be used to filter the record displays in the list of the lookup, you need to specify the field name in the “filterkeys” attribute. In case there are multiple fields set in the “filterkeys” attribute, it is recommended that you have a placeholder text in the search field so that you can enter the text in the “placeholder” attribute.

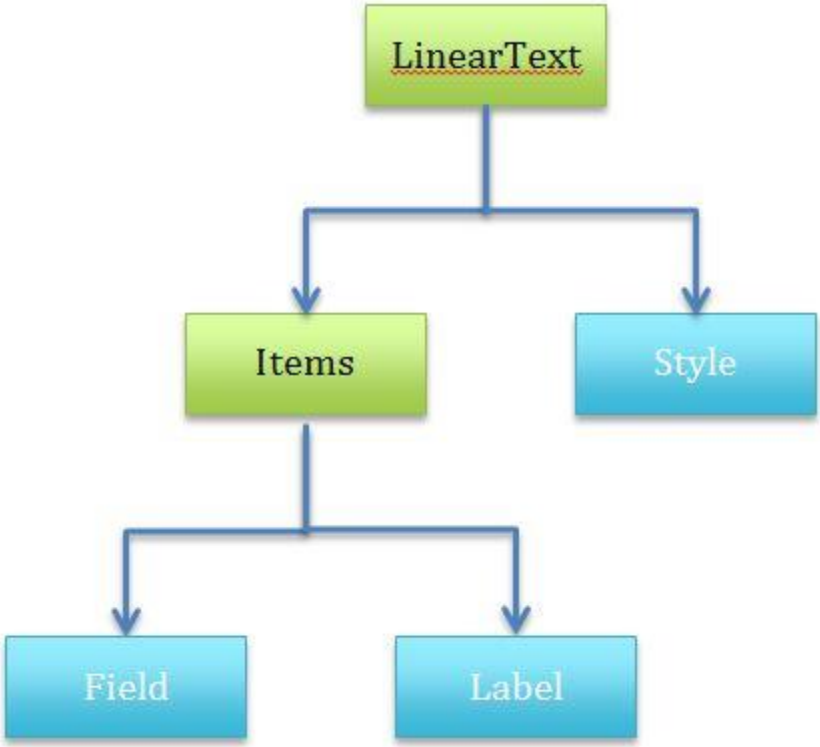
Rules

- The “filterkeys” attribute is mandatory for the searchfield component.
- Setting multiple fields in the “filterkeys” should be separated by comma character (for example, “KeyValue, DisplayValue”).
- The value that can be set in “filterkeys” is dependent on the response of the “search” operation or the fields being returned by the search layout. (Refer to the DeltekTouch.I).
- The “placeholder” attribute is optional.
- This component must only be used in the lookup layout file.

Examples

```
{
      region: #N”SearchFieldArea”,
      protectedregion: false,
      type: #N“searchfield”,
      filterkeys: #N”KeyValue, DisplayValue”,
      placeholder: #T”Task Name, Task Description”
}
```

LinearText



Description

This component is used to group components, such as **Label** and **Field**, and display them in a single line. It can only group **Label** and **Field**.

Rules

- If there is a “style” defined in the “lineartext” component, this means that it applies to all items. For example, the style contains “fontsize:14px” and the items of the “lineartext” contained two (2) labels and one (1) field; those will have a 14px of font size.
- If the styling in the item is present, it overrides the styling of the “lineartext” component.
- You need to declare the “items” attribute for the component.
- The “style” attribute is optional.
- Always append the “card” prefix to the source to let the code know that this source needs to be extracted on the card part (header screen). This code, however, only to the Approvals screen.

Examples

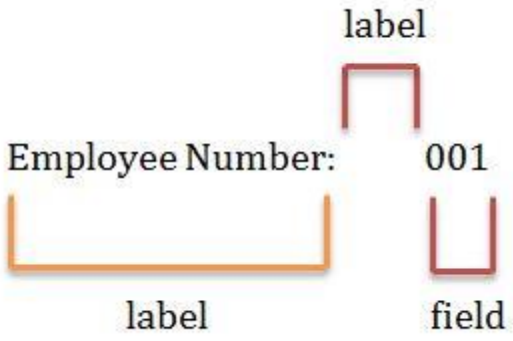
The following example is a simple way to declare a “lineartext” component in the layout file:

```
{
        type: “lineartext”,
        items: [
                {
                        type: “label”,
                        title: #T”Employee Number:”
                },
                {
                        type: “label”,
                        title: #T“          ”
                },
                {
                        type: “field”,
                        source: #N”EmployeeNumber”
                }
        ]
}
```

Note: There are 10 spaces added in the following line, between “and”:

```
Title: #T“          ”
```

Output

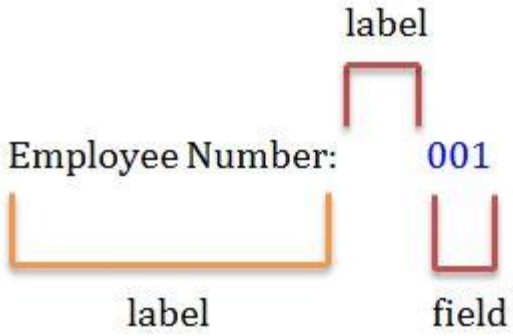


To apply “style” to the “lineartext” component (using the example above) by making the “field” item in blue, see the following example:

```
{
        type: “lineartext”,
```

```
        style: {
                fontsize: "14px",
                fontweight: "bold"
        },
        items: [
                {
                        type: "label",
                        title: #T"Employee Number:"
                },
                {
                        type: "label",
                        title: #T"            "
                },
                {
                        type: "field",
                        source: #N"EmployeeNumber",
                        style: {
                                color: "blue"
                        }
                }
        ]
}
```

Output

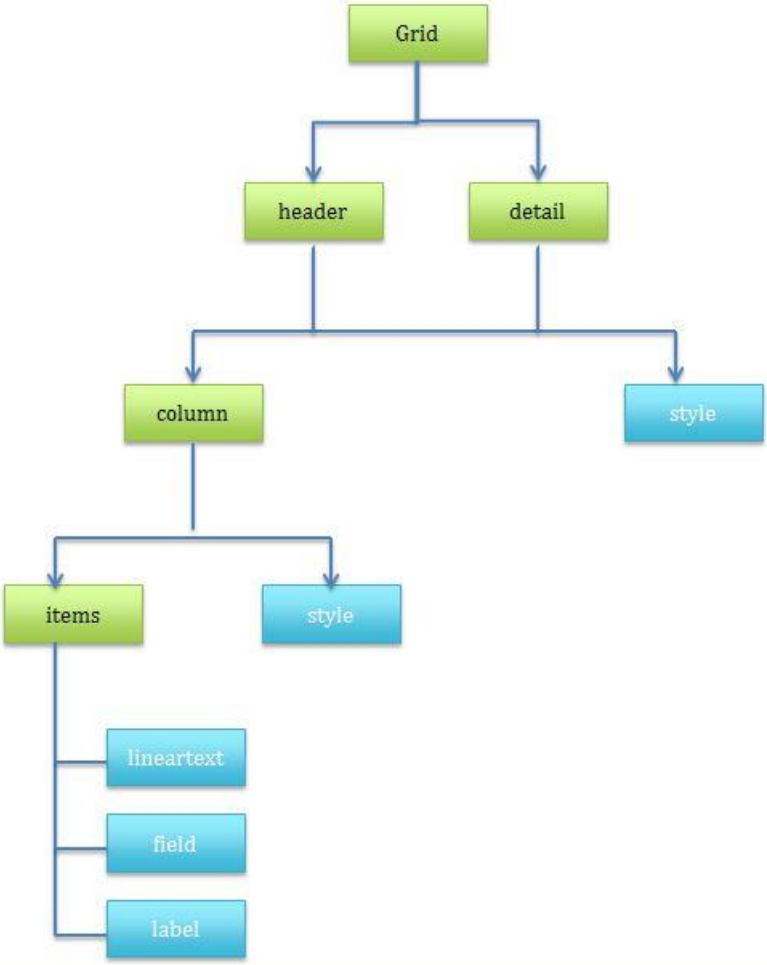


Use the following code to display data on the card part (header screen):

```
{
        type: #N"field",
```

```
        source: #N"card.name1"
}
```

Grid

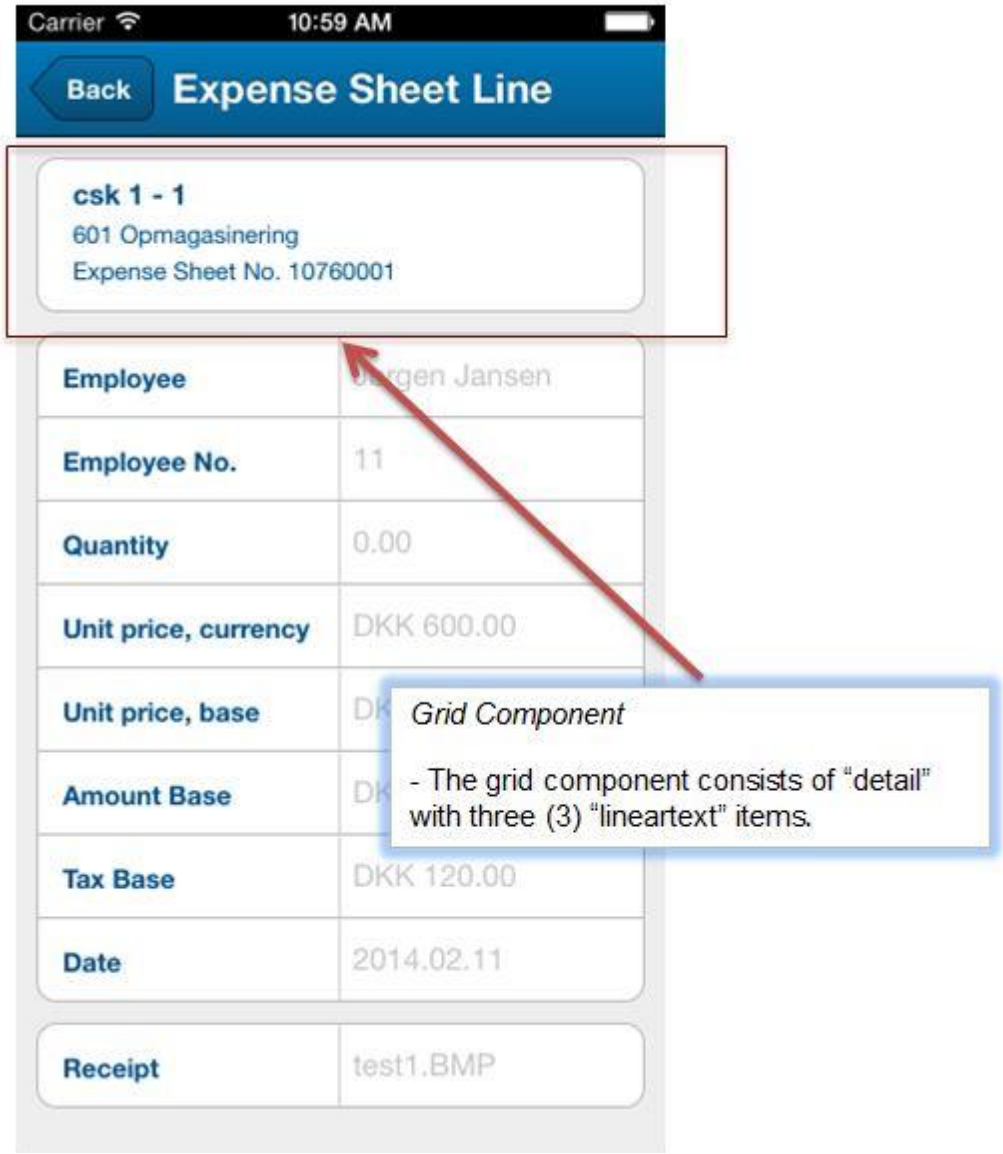


Description

This component is mainly used to display the data in a tabular representation. The “header” and “detail” are the main attributes of this component. Not all of “grid” in the layout file contains the “header” attribute. You only need the said attribute if the control you are targeting to customize on a screen contains the column title (for example, the **Summary** screen).

Grid is one of the very useful components in the layout customization. It is also being used on the “Line Screens” if one of the requirements is to group a set of fields in one section of the screen.

Figure 2 Sample Line Screen: Expense Sheet Line



The "grid" component can support multiple columns, but there are some screens you should not apply any additional columns because some of the portions in the grid component are not customizable (for example, the**Timesheet** screen). You can try to customize this, but doing so displays a bad representation of data on the screen.

Rules

- Do not specify the "header" or "detail" attributes if the particular layout file do not contain any of these attributes. Doing so might cause some problematic rendering of the layout on the screen or Deltek Touch would not be able to read it.
- The "header" or "detail" attributes must contain at least one "column" component.

- By default, if there are two (2) “column” specified in the “header” or “detail” attributes, the first column width occupies the 70% width of the screen and the second column 30% width of the screen.
- If there are more than two (2) columns, the width is divided evenly.
- Most of the styling aspects (for example, Font Size and Font Color) in the “header” attribute are built-in in the Touch Application.

Examples

The following example is a general syntax declaration of the “grid” component that consists of the “header” and “detail” attributes:

```
{
        type: “grid”,
        header: [
                {
                        type: “column”
                },
                {
                        type: “column”
                }
        ],
        detail: [
                {
                        type: “column”
                },
                {
                        type: “column”
                }
        ]
}
```

The following example is a general syntax declaration of the “grid” component that consists of the “detail” attribute:

```
{
        type: “grid”,
        detail: [
                {
                        type: “column”
                },
                {
```

```
                    type: "column"
                }
        ]
}
```

Complete syntax declaration of the “grid” component that consists of the “header” and “detail” attributes.

```
        }
        ],
        detail: [{
                    type: "column",
                    items: [{
                                type: "lineartext",
                                style: {
                                        fontsize: "14px",
                                        color: "#00488a"
                                },
                                items: [
                                        {
                                                type: "field",
                                                source: "ShortDate"
                                        },
                                        {
                                                type: "label",
                                                title: " - "
                                        },
                                        {
                                                type: "field",
                                                source: "WeekDay"
                                        }
                                ]
                            }
                    ]
                },
                {
                    type: "column",
                    style: {
                            textalignment: "right",
```

```
                        fontsize: "14px",
                        paddingright: "10px",
                        color: "#00488a",
                        fontweight: "bold"
                    },
                    items: [ {
                                type: "field",
                                source: "TotalHoursRegisteredHours",
                                mask: "hoursnozeroes"
                        }
                    ]
                }
        ]
}
```

Output

Here is another example of the “grid” component syntax declaration that consists of the “detail” attribute, wherein the second column is not customizable.

```
{
      type: “grid”,
      detail: [{
                  type: “column”,
                  items: [{
                              type: “lineartext”,
                              style: {
                                    fontsize: “14px”,
                                    color: “004488a”,
                                    fontweight: “bold”
                              },
                              items: [{
                                          type: “field”,
                                          source: “JobNameVar”
                                    },
                                    {
                                          type: “label”,
                                          title: “ – “,
                                    },
                                    {
                                          type: “field”,
                                          source: “JobNumber”
                                    }
                              ]
                        },
                        {
                              type: “lineartext”,
                              style: {
                                    fontsize: “12px”
                                    color: “#6b6b6b”
                              },
                              items: [{
                                          type: “field”,
                                          source: “TaskName”
```

```
                                        },
                                        {
                                                type: "label",
                                                title: " – "
                                        },
                                        {
                                                type: "field",
                                                source: "TaskDescriptionVar"
                                        }
                                ]
                        },
                        {
                                type: "lineartext",
                                style: {
                                        fontsize: "12px",
                                        color: "#6b6b6b"
                                },
                                items: [
                                        {
                                                type: "field",
                                                source: "CustomerNameVar"
                                        }
                                ]
                        },
                        {
                                type: "lineartext",
                                style: {
                                        fontsize: "12px",
                                        color: "#6b6b6b"
                                },
                                items: [
                                        {
                                                type: "field",
                                                source: "Invoiceable",
                                                mask: {
                                                        Description: {
```

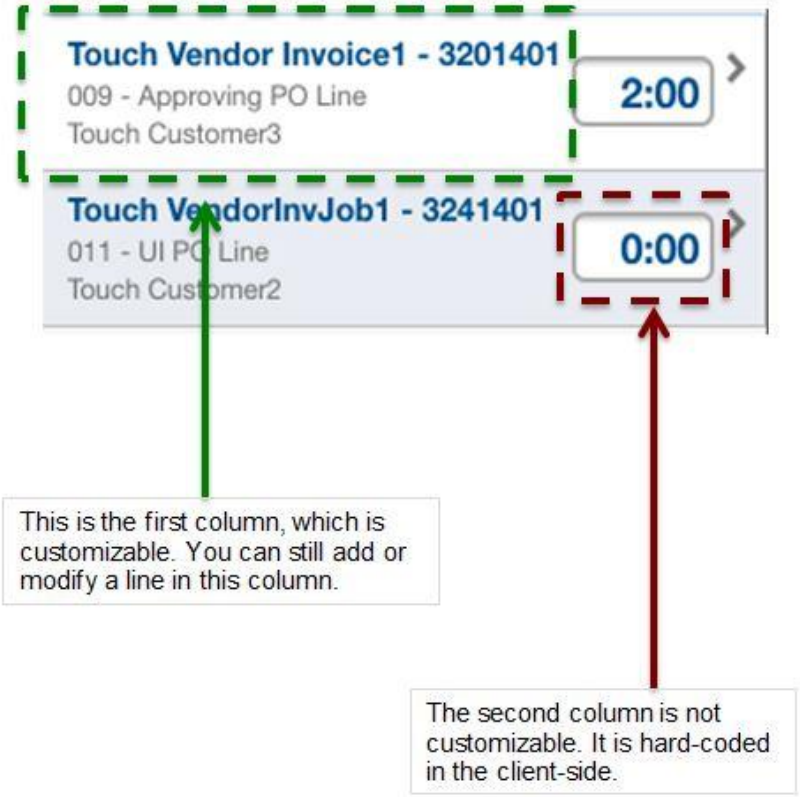
```
                                                                    truevalue: "Billable",
        falsevalue: "Non-Billable"
                                                                  }
                                                                }
                                                              }
                                                            ]
                                                          }
                                                        ]
                                                      },
                                                      {
                                                        type: "column",
                                                        region "HourField",
                                                        protectedregion: true
                                                      }
                                                    ]
                                                  }
```

Note: The following last three lines refer to the second column, but it is not customizable. The Touch client handles the code and display of the hour field on the Timesheet screen. In short, it is hard-coded in the client-side:

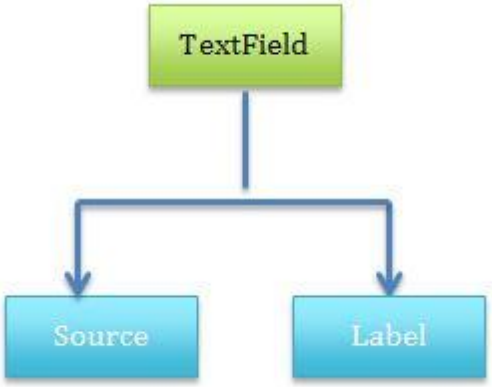
```
{
                    type: "column",
                    region "HourField",
                    protectedregion: true
              }   "
```

Output

The following is a portion of layout on the **Timesheet** screen:



TextField



Description

This component is used to display database field and label on Line screens. For example, **Expense Sheet Line**:

Rules

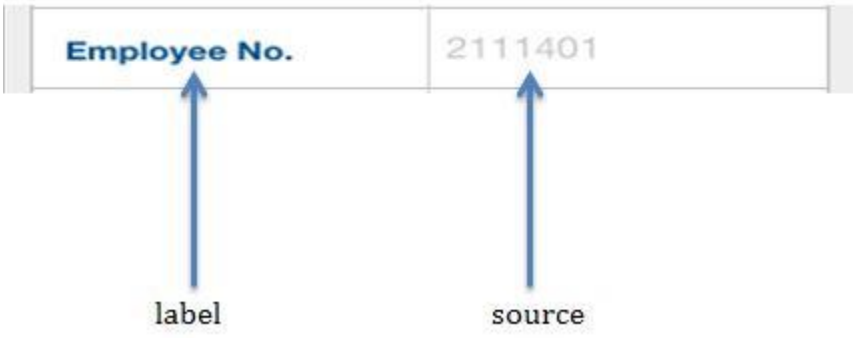
- Always specify the “source” and “label” attributes and their corresponding values.
- This component does not support styling. The formatting of the font used in a label is provided automatically by the client-side.
- Always append the “card” prefix to the source to let the code know that this source needs to be extracted on the card part (header screen). This code, however, only to the Approvals screen.

Examples

This is a general syntax declaration of the “textfield” component:

```
{
        type: “textfield”,
        source: “EmployeeNumber”,
        label: “Employee No.”
}
```

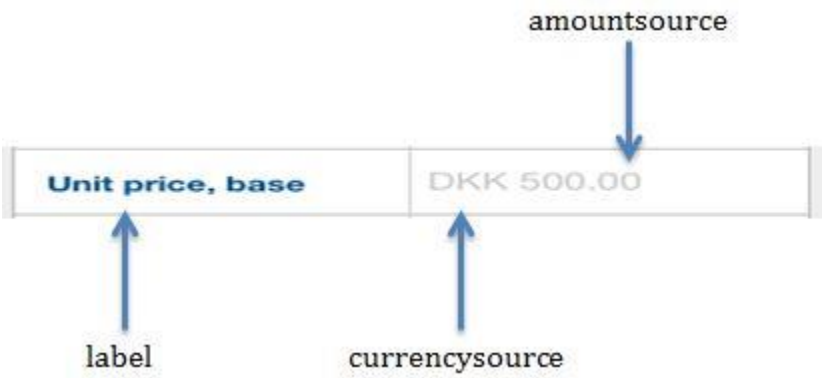
Output



If there is a currency and an amount field that need to display in a single “textfield” component, use the “mask” attribute.

```
{
        type: “textfield”,
        source: “UnitPriceCurrency”,
        label: “Unit price, currency”,
        mask: {
                Description: {
                        currencysource: “Currency”,
                        amountsource: “UnitPriceCurrency”
                }
        }
}
```

Output



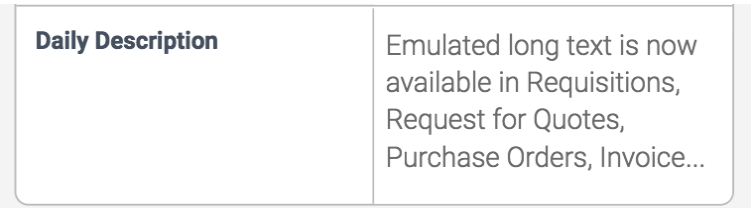
Use the following code to display data on the card part (header screen):

```
{
        type: #N"textfield",
        source: #N"card.requisitioneremployeenamevar",
        label: #T"Requisitioner"
}
```

TextareaField

Description

This component is used to display a text area field on a Line screen. It defines a multi-line text input control and can hold an unlimited number of characters depending on the system setup. For example, **Timesheet Line**:



Rules

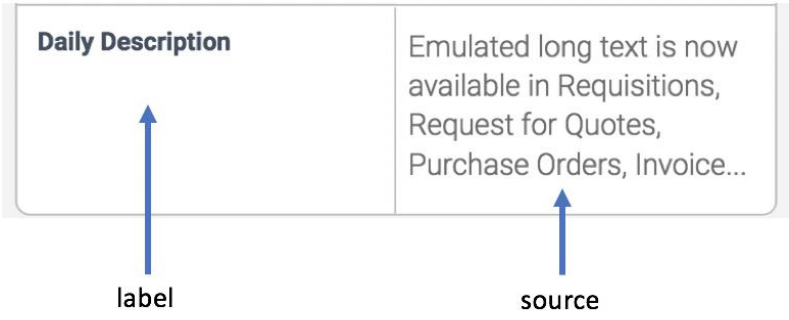
- Always specify the “source” and “label” attributes and their corresponding values.
- This component does not support styling. The formatting of the font used in a label is provided automatically by the client-side.
- The default maximum length of this component is set to **255** characters. You need to change DeltekTouch.I to enable support long text fields.

Examples

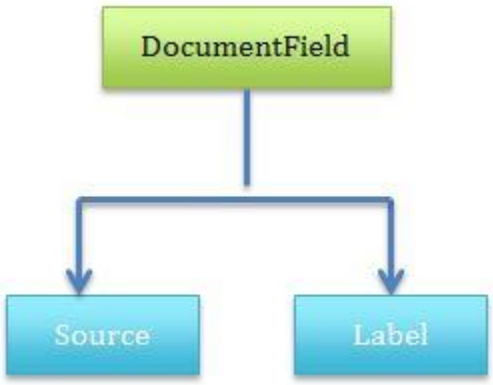
This is a general syntax declaration of the “textareafield” component:

```
{
        type: "textareafield",
        source: "Daily Description",
        label: "Daily Description"
}
```

Output



DocumentField



Description

This component is used for displaying attached document on a separate screen once a user taps a field (for example, PDF, PNG, or IMG files).

Rules

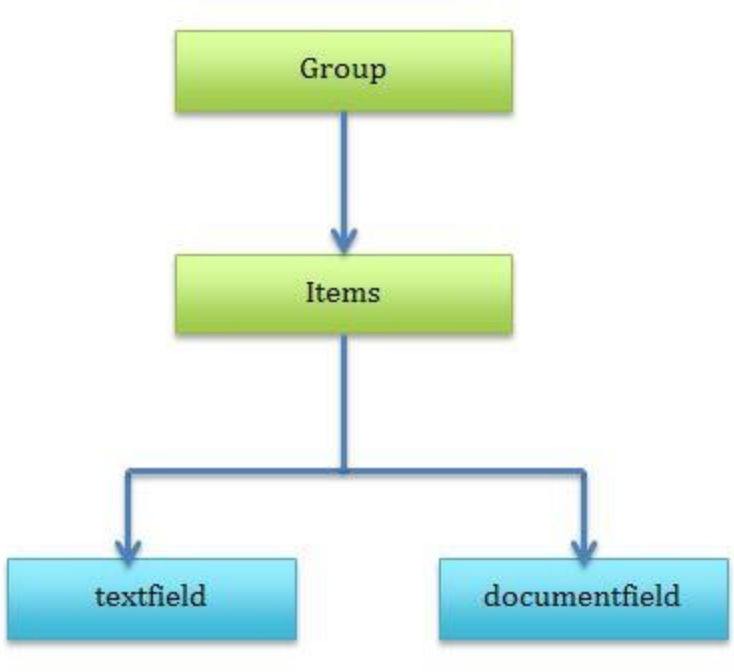
- Always specify the “source” and “label” attributes as well as the corresponding values.
- The “documentfield” component only works properly if the “source” attribute contains “DocumentName” value (see image below).

```
{
    type: #N"group",
    items: [
        {
            type: #N"documentfield",
            source: #N"DocumentName",
            label: #T"Receipt"
        }
    ]
}
```

- The “mask” attribute is not applicable in this component.
- This component does not support styling. The formatting of the font of a label is provided automatically by the client-side.
- The separate screen that displays the attached document is not customizable (see Figure 4 below).

Figure 3: Preview Screen

Group



Description

This component is used to group a set of fields into one component.

Rules

- This component does not support any styling.
- Only “textfield” and “documentfield” can be set in this component.
- This is a required component if you need to add a single or multiple “textfield” or “documentfield” components.

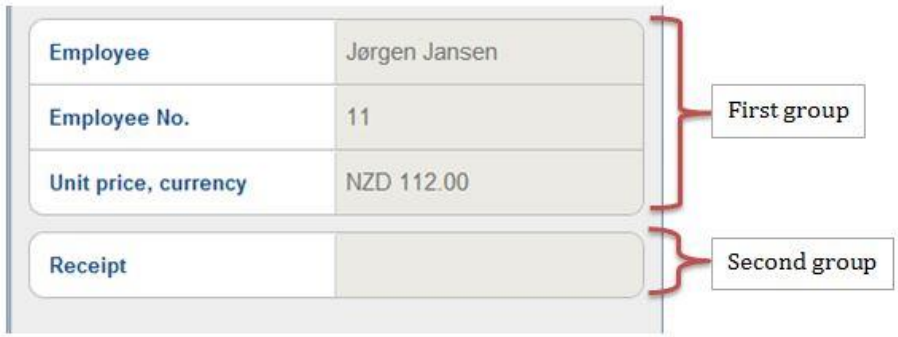
Example

This is a general syntax declaration when using the “group” component.

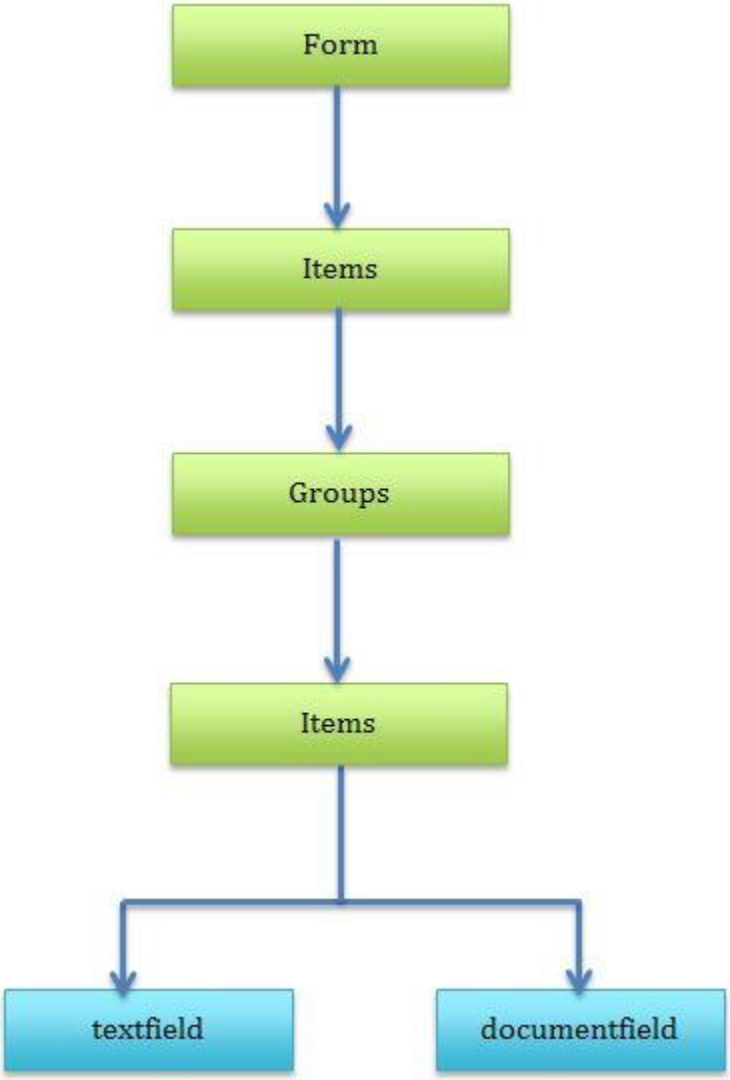
```
{
        type: “group”,
        items: [{
                        type: “textfield”,
                        source: “EmployeeNameVar”,
                        label:  “Employee”
                },
                {
```

```
                    type: "textfield",
                    source: "EmployeeNumber",
                    label: "Employee No."
            },
            {
                    type: "textfield",
                    source: "UnitPriceCurrency",
                    label: "Unit price, currency",
                    mask: {
                          Description: {
                                currencysource: "Currency",
                                amountsource: "UnitPriceCurrency"
                          }
                    }
            }]
},
{
      type: "group",
      items: [{
                    type: "documentfield",
                    source: "DocumentName",
                    label: "Receipt"
            }]
}
```

Output



Form



Description

This component is used to tell the Touch Client that the component it is encapsulating in the “items” attribute is designed for “Line Screens.” Typically, it is using the HTML5 or Sencha Touch native controls (for example, textfield).

Rules

- This component does not support any styling.
- Always defined a “group” component in the “items” attribute.
- You cannot set a “textfield” or a “documentfield” directly in the “items” attribute.

- You cannot set a “form” component inside the “items” attribute.

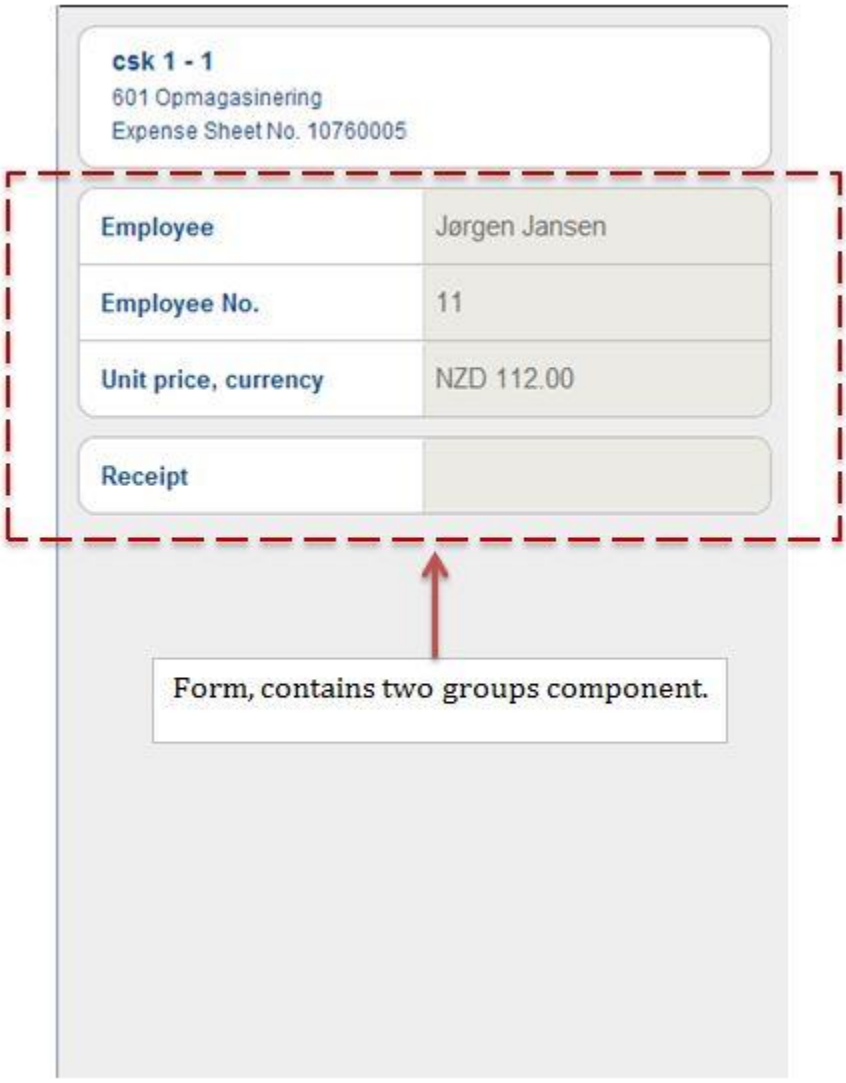
Example

This is a general syntax declaration when using the “form” component.

```
{
        type: “form”,
        items: [{
              type: “group”,
              items: [{
                          type: “textfield”,
                          source: “EmployeeNameVar”,
                          label:  “Employee”
                    },
                    {
                          type: “textfield”,
                          source: “EmployeeNumber”,
                          label: “Employee No.”
                    },
                    {
                          type: “textfield”,
                          source: “UnitPriceCurrency”,
                          label: “Unit price, currency”,
                          mask: {
                                Description: {
                                      currencysource: “Currency”,
                                      amountsource: “UnitPriceCurrency”
                                }
                          }
                    }]
              },
              {
                    type: “group”,
                    items: [{
                          type: “documentfield”,
                          source: “DocumentName”,
                          label: “Receipt”
```

```
                }]
        }
]}
```

Output



Attachment Icon

Description

This component is used for displaying an icon that represents “attachment.”

Rules

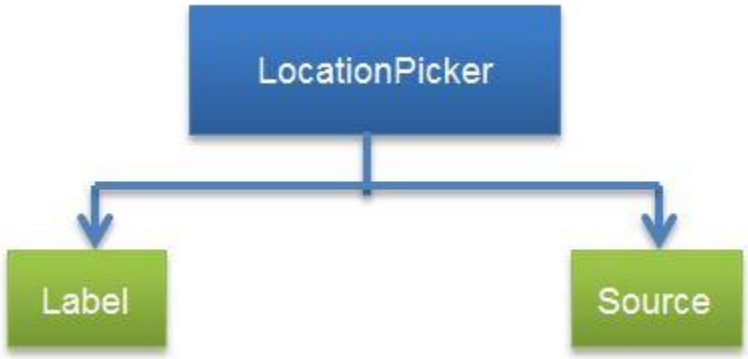
- This component should be used inside a column component.
- This component does not contain any attributes.

Examples

This is a general syntax declaration of the “attachmenticon” component:

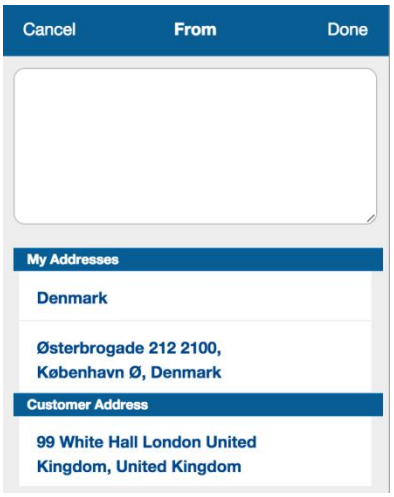
```
{
        type: “column”,
        items: [
                {
                        type: “attachmenticon”
                }
        ]
}
```

Location Picker



Description

This component is used for entering and displaying addresses.



Rules

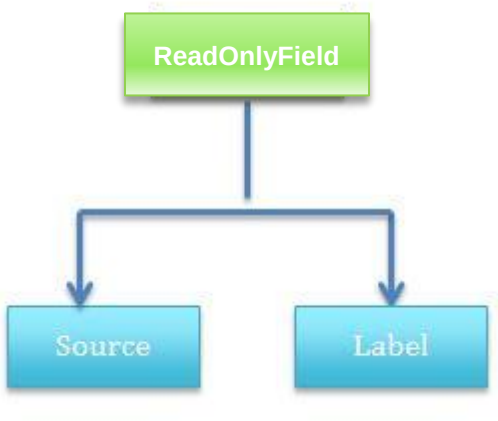
This component does not support any styling.

Examples

This is a general syntax declaration of the “locationpicker” component:

```
{
        source: “MileageFrom”,
        label: “From”,
        type: “locationpicker”
}
```

ReadOnlyField



Description

This component is used to display database field and label on the **Form** and **Line** screens that cannot be modified. For example, on the **Purchase Order** form:

Submitted	No

Rules

- Always specify the “source” and “label” attributes as well as the corresponding values.
- This component does not support styling. The formatting of the font used in a label is provided automatically by the client-side.

Examples

This is a general syntax declaration of the “readonlyfield” component:

```
{
        type: #N"readonlyfield",
        source: #N"submitted",
        label: #T"Submitted",
        mask: #N"yesno"
}
```

Output

- If the “Submitted” field value is “true,” the output is YES.
- If the “Submitted” field value is “false,” the output is NO.

Conditional Function Helpers

Description

The following functions are used to determine whether a field is available in the REST data response:

- this.defined
- this.noDefined

The layouts in Deltek Touch can be applied on several Maconomy versions (for example, same layout is applied on Maconomy 2.4 LA 1, 2.4 LA 2, 2.4 GA, and 2.4 .x). Some Maconomy versions, however, may contain fields that are not available in other Maconomy versions (for example, the **timeregistrationunit** field is available in Maconomy 2.4 GA but not in Maconomy 2.4 LA 1/ LA 2).

If you want to use such fields in the layouts, you need to check if the field is supported in the Maconomy version to which Deltek Touch is connecting using these functions.

Rules

You cannot use these functions in your custom layouts yet. You can only apply the said functions in the following standard layouts:

- AbsenceApprovalFormLayoutREST.l
- absenceRequestLayoutREST.l
- AllowanceApprovalFormLayoutREST.l
- AllowanceApproveInputScreenREST.l
- allowanceRequestLayoutREST.l

Example

```
{
        if: [#N"this.defined('timeregistrationunit') &&
        (this.notDefined('timeregistrationunitprevcurrnextvar') ||
        timeregistrationunitprevcurrnextvar == 'nil')", {
        label: #T"Time Unit",
        source: #N"timeregistrationunit",
        type: #N"selectfield"
         }]
 },
```

Metadata Files

Each layout has a corresponding metadata file. The metadata and layout files have the same name, but the former ends with Metadata.I while the latter is Layout.I (for example, **TimesheetMetadata.I** and **TimesheetLayout.I**).

Note: The metadata file is for informational purposes only. Deltek Touch does not use these files. For light-weight customizations, it does not make sense to customize the metadata files.

The metadata file contains the list of fields and variables you can use in the corresponding layout. This file is relevant, if you want to perform any of the following tasks:

- Add fields to the layout
- Change fields in the layout
- Find out which fields or variables you can you add to the layout

Note: In case your layout customization is only about formatting, the metadata file is irrelevant.

The information in the metadata file is structured similarly to the layout in regions. For each region, you have the list of fields or variables you can use, starting with the ones included in the standard layout.

In some cases, the metadata contains additional comments related to the source of the fields or variables.

Example

```
// Fields from TimeSheetLine
// Variables from SpeedSheet dialog, table part
```

If you want to show additional data on a Deltek Touch screen, you should check if the data is available in the metadata.

- If the data is in the metadata, you can add it to the layout. This process is called light-weight customization.
- If the data is not in the metadata, you need to customize the Web service to include this data first, extend the metadata, and, finally, change the layout. This process is called heavy-weight customization, which is outside the scope of this document.

Example

Show the Location dimension on Time Sheet lines. To do this, perform the following steps:

1. Find the **Timesheet** screen.
2. Find the **TimesheetLayout.I** layout file.
3. Find a region in the layout, where you want to add the Location dimension: **RegistrationList**
4. Find the **TimesheetsMetadata.I** metadata file.
5. Find the Location dimension in the metadata, in the RegistrationList region:

```
{region: #K"RegistrationList",
    operation: #K"getperiod",
```

```
metadata:
   [
      // Fields from DailyTimeSheetLine
      {source: #K"ActivityNumber"},
      {source: #K"TaskName"},
      {source: #K"NumberOf"},
      {source: #K"DailyDescription"},
      {source: #K"EmployeeNumber"},
      {source: #K"TheDate"},
      {source: #K"EntryText"},
      {source: #K"ActivityType"},
      {source: #K"NumberTransferred"},
      {source: #K"Remark"},
      {source: #K"Description"},
      {source: #K"InternalJob"},
      {source: #K"LocationName"},
      {source: #K"EntityName"},
      {source: #K"ProjectName"},
      …
```

6. Change the layout by adding **LocationName** to the **RegistrationList** region in **TimesheetLayout.l**.

Appendix A: Screens, Layouts, and Metadata Files

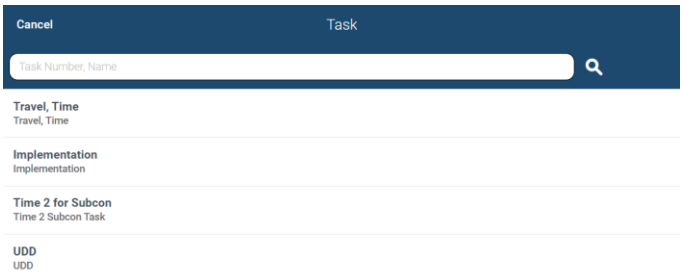
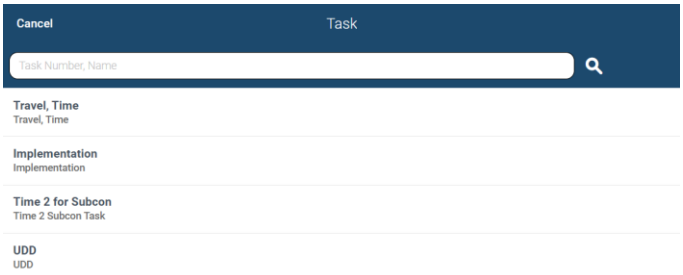
The following sections display the customizable Deltek Touch screens, and the corresponding layout and metadata files to modify.

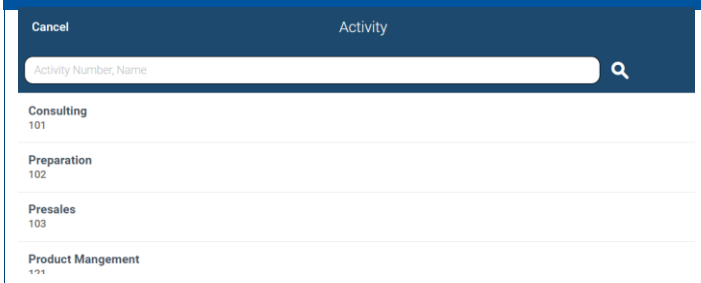
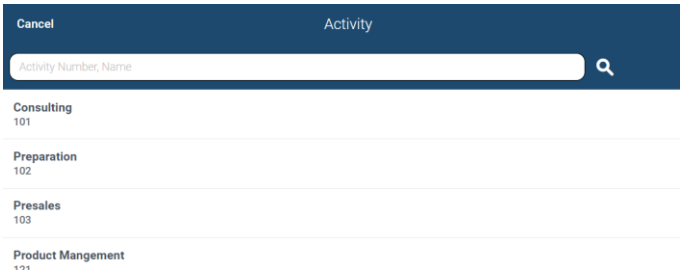
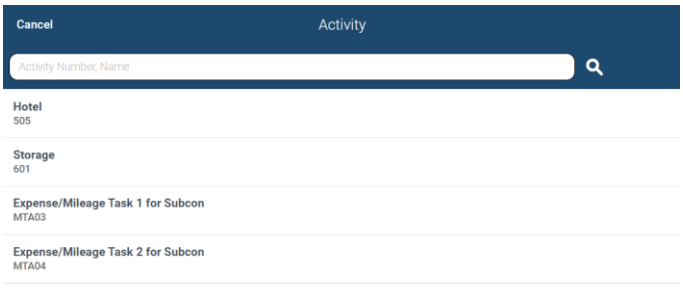
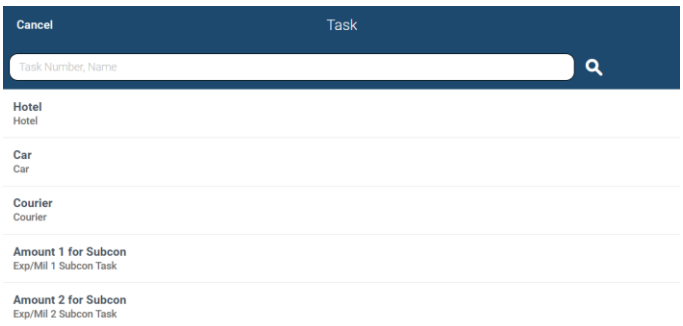
Note: In case your layout customization is only about formatting, the metadata file is irrelevant.

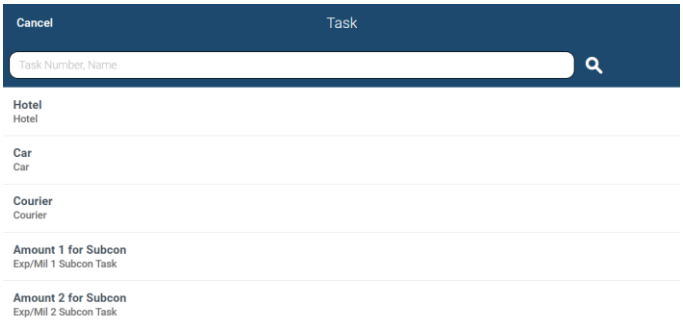
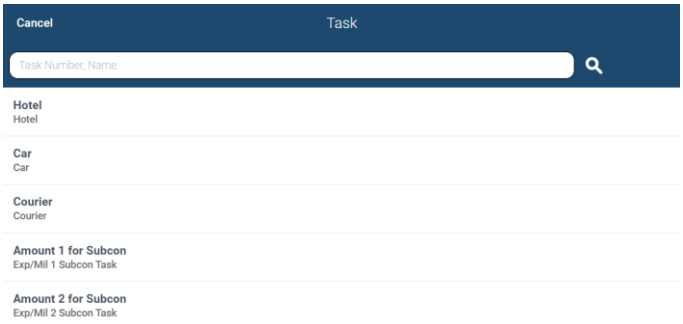
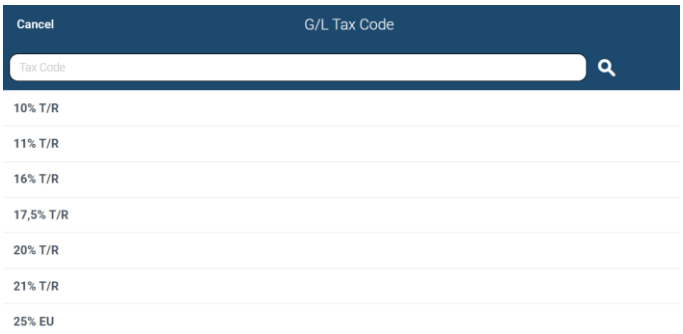
Search Screens

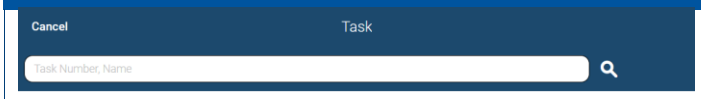
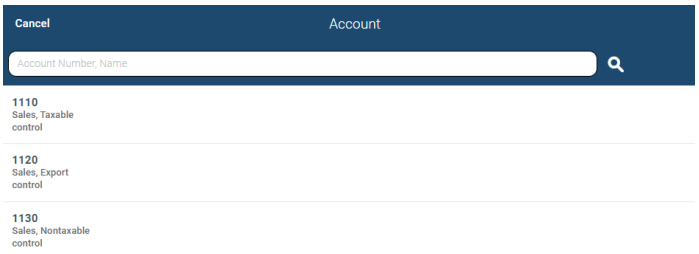
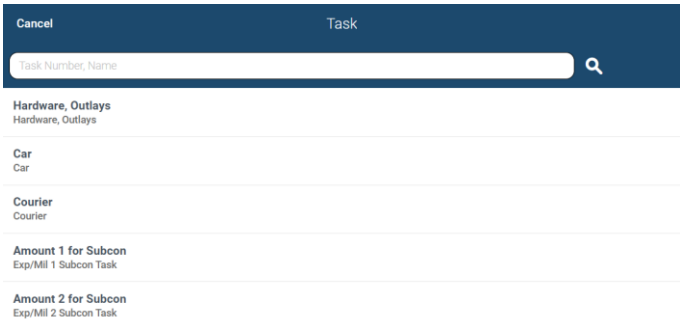
Starting with Deltek Touch 3.6, you can customize basic and advanced search screens by appending your own condition in REST Touch layouts.

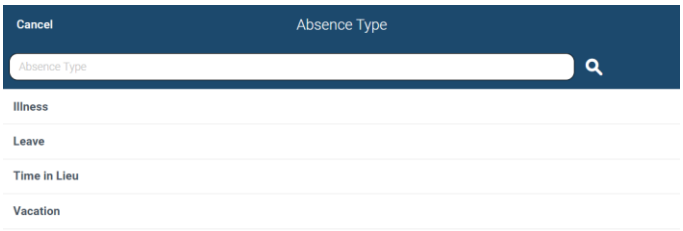
Basic Search Screens

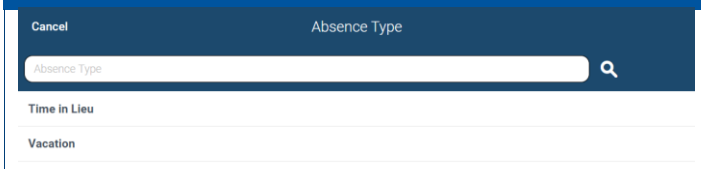
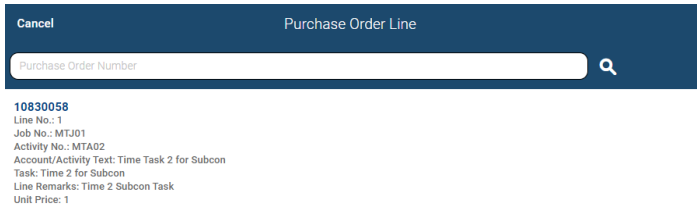
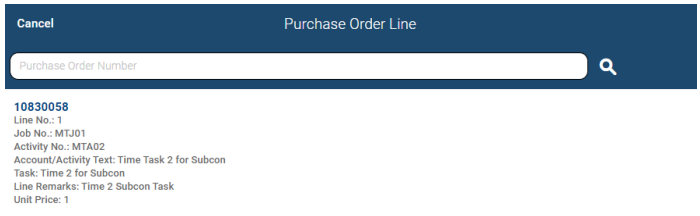
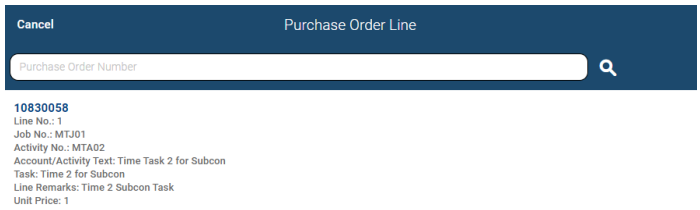
Navigation Path and Screens	REST
Timesheet » Timesheet Line » Task (Daily)  	**Screen Layout:** TimesheetLogEntryDailyLayoutREST.I **Search Layout:** Find_TaskListLine.I **Container:** Find_TaskListLine
Timesheet » Timesheet Line » Task (Weekly)  	**Screen Layout:** TimesheetLogEntryWeeklyLayoutREST.I **Search Layout:** Find_TaskListLine.I **Container:** Find_TaskListLine
Timesheet » Timesheet Line » Activity (Daily)	**Screen Layout:** TimesheetLogEntryDailyLayoutREST.I **Search Layout:**

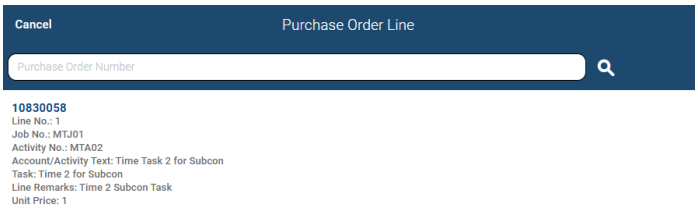
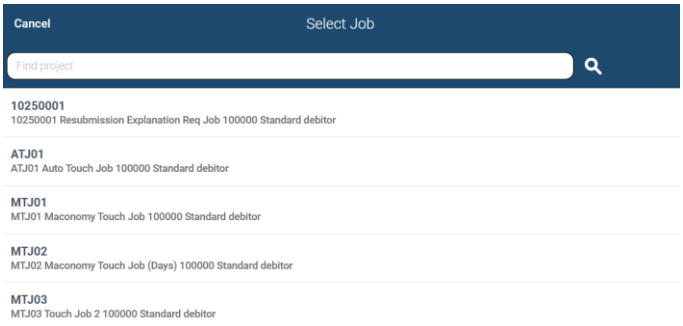
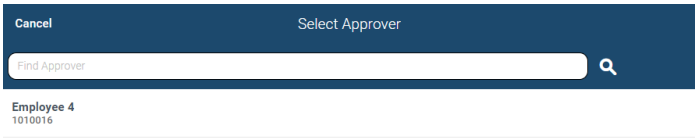
Navigation Path and Screens	REST
	Find_Activity.I **Container:** Find_Activity
Timesheet » Timesheet Line » Activity (Weekly) 	**Screen Layout:** TimesheetLogEntryWeeklyLayoutREST.I **Search Layout:** Find_Activity.I **Container:** Find_Activity
Expense Sheet » Expense Sheets » Expense Sheet Lines » Activity 	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** Find_Activity.I **Container:** Find_Activity
Expense Sheet » Expense Sheets » Expense Sheet Lines » Task 	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** Find_TaskListLine.I **Container:** Find_TaskListLine

Navigation Path and Screens	REST
Quick Capture » Attach Receipt » Create Expense Sheet Line  	**Screen Layout:** QuickCaptureExpenseSheetLineLayoutREST.I **Search Layout:** Find_TaskListLine.I **Container:** Find_TaskListLine
Quick Capture » Select Expense Sheet » Expense Sheet Line  	**Screen Layout:** QuickCaptureICRExpenseSheetLineLayoutREST.I **Search Layout:** Find_TaskListLine.I **Container:** Find_TaskListLine
Expense Sheet » Expense Sheets » Expense Sheet Lines » Tax Code  	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** Find_FinanceVATCode.I **Container:** Find_FinanceVATCode
Mileage Sheets » Mileage Sheet » Mileage Sheet Line » Task	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** Find_TaskListLine.I

Navigation Path and Screens	REST
 No record found	**Container:** find_tasklistline
Purchase Order » Purchase Orders » Requisitioner 	**Screen Layout:** PurchaseOrderLayoutREST.I **Search Layout:** Find_Employee.I **Container:** Find_Employee
Purchase Order » Purchase Orders » Purchase Order » Purchase Order Lines » Account No. 	**Screen Layout:** PurchaseOrderLineLayoutREST.I **Search Layout:** Find_Account.I **Container:** Find_Account
Purchase Order » Purchase Orders » Purchase Order Lines » Task 	**Screen Layout:** PurchaseOrderLineLayoutREST.I **Search Layout:** Find_TaskListLine.I **Container:** Find_TaskListLine

Navigation Path and Screens	REST
Purchase Order » Purchase Orders » Purchase Order Lines » Activity 	**Screen Layout:** PurchaseOrderLineLayoutREST.I **Search Layout:** Find_Activity.I **Container:** Find_Activity
Purchase Order » Purchase Orders » Purchase Order Lines » Employee 	**Screen Layout:** PurchaseOrderLineLayoutREST.I **Search Layout:** Find_Employee.I **Container:** Find_Employee
Absence Requests » Absence Request » Absence Type  	**Screen Layout:** AbsenceRequestLayoutREST.I **Search Layout:** Find_AbsenceType.I **Container:** find_absencetype
Allowance Requests » Allowance Request » Absence Type	**Screen Layout:** AllowanceRequestLayoutREST.I **Search Layout:** Find_AbsenceType.I

Navigation Path and Screens	REST
	**Container:** Find_AbsenceType
Timesheet » Timesheet Line » Purchase Order (Daily) **Timesheet » Timesheet Line » Purchase Order Line** (Daily) 	**Screen Layout:** TimesheetLogEntryDailyLayoutREST.I **Search Layout:** Find_PurchaseOrderLine_Timesheet.I **Container:** Find_PurchaseOrderLine
Timesheet » Timesheet Line » Purchase Order (Weekly) **Timesheet » Timesheet Line » Purchase Order Line** (Weekly) 	**Screen Layout:** TimesheetLogEntryWeeklyLayoutREST.I **Search Layout:** Find_PurchaseOrderLine_Timesheet.I **Container:** Find_PurchaseOrderLine
Mileage Sheet » Mileage Sheets » Mileage Sheet Line » Purchase Order **Mileage Sheet » Mileage Sheets » Mileage Sheet Line » Purchase Order Line** 	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** Find_PurchaseOrderLine_ExpenseSheet.I **Container:** Find_PurchaseOrderLine

Navigation Path and Screens	REST
Expense Sheet » Expense Sheets » Expense Sheet Lines » Purchase Order **Expense Sheet » Expense Sheets » Expense Sheet Lines » Purchase Order Line**  	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** Find_PurchaseOrderLine_ExpenseSheet.I **Container:** Find_PurchaseOrderLine
Expense Sheet » Expense Sheets » Expense Sheet Lines » Justification	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_TheOption_ExpenseJustification.I **Container:** Find_TheOption
Approvals » Items for Approval » Select Job  	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_AllJobHeader_Approval.I **Container:** Find_JobHeader
Approvals » Items for Approval » Select Approver  	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_AllEmployee_Approval.I

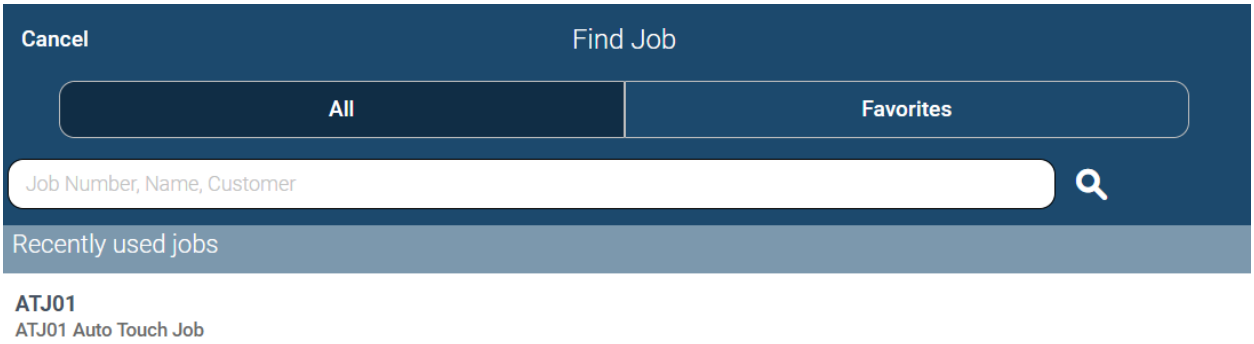
Navigation Path and Screens	REST
	Container: Find_JobHeader

Advanced Search Screens

The **Find Job**, **Find Vendor**, and **Address** screens are advanced lookup screens you can customize.

Find Job Screens

The **Find Job** screen allows you to look up your jobs, all existing jobs, any favorites, and even recently used jobs. You can modify the **Find Job** screen from the **Timesheet**, **Expense Sheet**, **Mileage Sheet**, and **Purchase Order** screens.



Navigation Path	REST
Timesheet » Timesheet Line » Find Job » All	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_JobHeader_Timesheet.l **Container:** Find_JobHeader
Timesheet » Timesheet Line » Find Job » Favorites	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_JobFavorite.l

Navigation Path	REST
	Container: Find_JobFavorite
Timesheet » Timesheet Line » Find Job » Recently Used Jobs	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_JobHeader_Timesheet.I **Container:** Find_TimesheetLineU
Timesheet » Timesheet Line » Find Job » My Jobs	**Screen Layout:** None, hard coded in the app **Search Layout:** Find_JobHeader_Timesheet.I **Container:** Find_JobEmployeeControlU
Expense Sheet » Expense Sheets » Find Job » All	**Screen Layout:** ExpenseSheetLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_JobHeader
Expense Sheet » Expense Sheets » Find Job » Recently Used Jobs	**Screen Layout:** ExpenseSheetLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:**

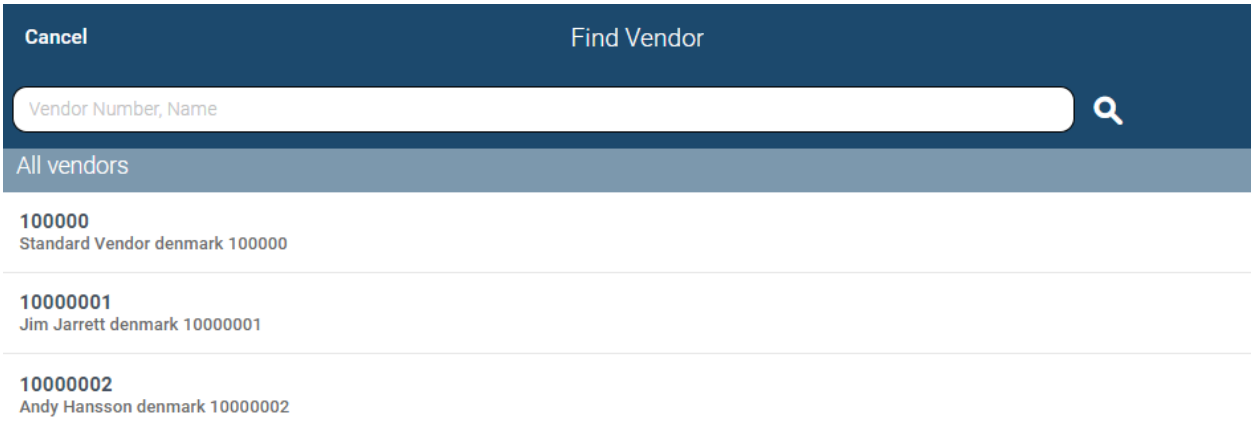
Navigation Path	REST
	Find_ExpenseSheetLineU
Expense Sheet » Expense Sheets » Expense Sheet Lines » Find Job » All	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_JobHeader
Expense Sheet » Expense Sheets » Expense Sheet Lines » Find Job » Favorites	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** Find_JobFavorite.I **Container:** Find_JobFavorite
Expense Sheet » Expense Sheets » Expense Sheet Lines » Find Job » Recently Used Jobs	**Screen Layout:** ExpenseSheetLineLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_ExpenseSheetLineU
Purchase Order » Purchase Orders » Purchase Order » Job » Find Job » All	**Screen Layout:** PurchaseOrderLayoutREST.I **Search Layout:** FindJobHeader_PurchaseOrder.I **Container:** Find_JobHeader

Navigation Path	REST
Purchase Order » Purchase Orders » Purchase Order » Job » Find Job » Recently Used Jobs	**Screen Layout:** PurchaseOrderLayoutREST.I **Search Layout:** FindJobHeader_PurchaseOrder.I **Container:** Find_PurchaseOrderLineU
Purchase Order » Purchase Orders » Purchase Order » Purchase Order Lines » Job » Find Job » All	**Screen Layout:** PurchaseOrderLineLayoutREST.I **Search Layout:** FindJobHeader_PurchaseOrder.I **Container:** Find_JobHeader
Purchase Order » Purchase Orders » Purchase Order » Purchase Order Lines » Job » Find Job » Recently Used Jobs	**Screen Layout:** PurchaseOrderLineLayoutREST.I **Search Layout:** FindJobHeader_PurchaseOrder.I **Container:** Find_PurchaseOrderLineU
Mileage Sheet » Mileage Sheet » Find Job » All	**Screen Layout:** MileageSheetLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_JobHeader

Navigation Path	REST
Mileage Sheet » Mileage Sheet » Find Job » Recently Used Jobs	**Screen Layout:** MileageSheetLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_ExpenseSheetLineU
Mileage Sheet » Mileage Sheet » Mileage Sheet Line » Find Job » All	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_JobHeader
Mileage Sheet » Mileage Sheet » Mileage Sheet Line » Find Job » Recently Used Jobs	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:** Find_ExpenseSheetLineU
Mileage Sheet » Mileage Sheet » Mileage Sheet Line » Find Job » Favorites	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** Find_JobFavorite.I **Container:** Find_JobFavorite

Find Vendor Screens

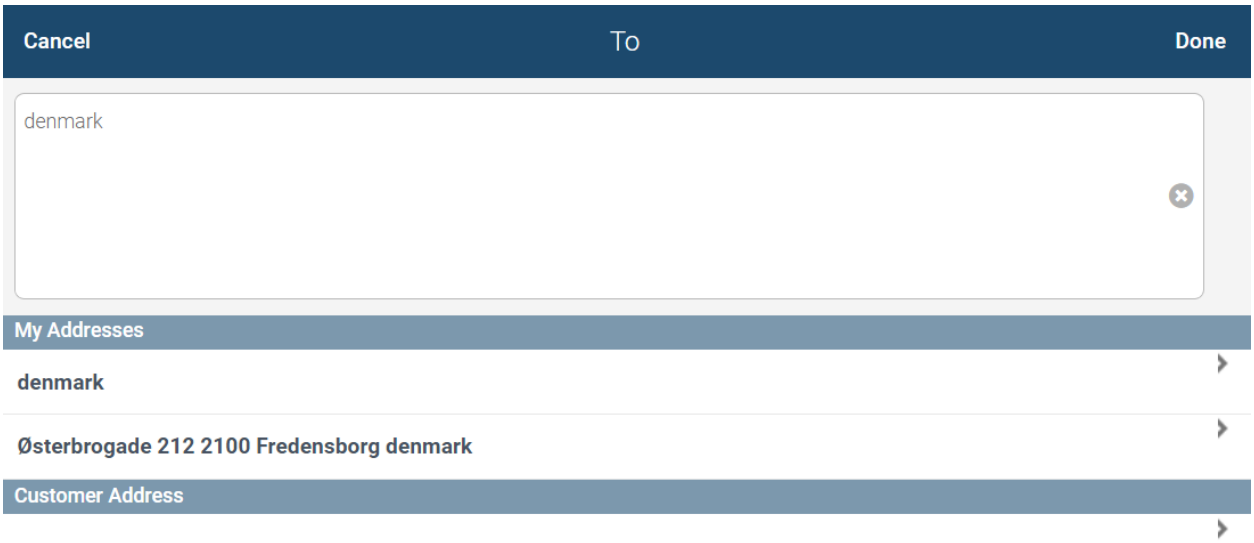
The **Find Vendor** screen allows you to look up vendors on the **Purchase Order** screen.

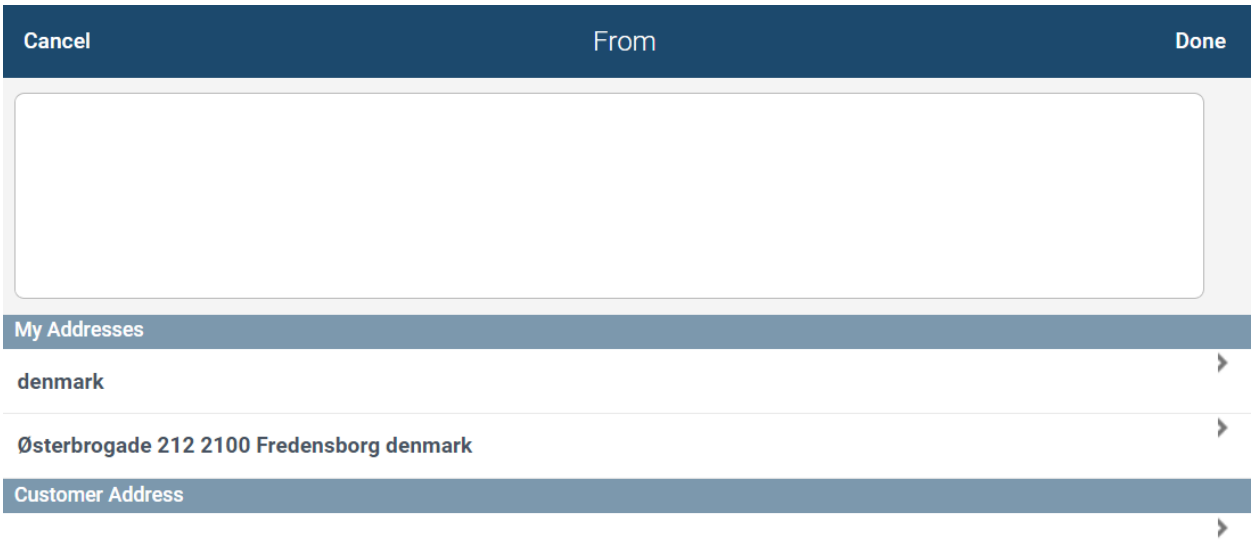


Navigation Path	REST
Purchase Order » Purchase Orders » Purchase Order » Find Vendor » Recently Used Vendor	**Screen Layout:** PurchaseOrderLayoutREST.I **Search Layout:** FindVendor_PurchaseOrder.I **Container:** PurchaseOrders
Purchase Order » Purchase Orders » Purchase Order » Find Vendor » All	**Screen Layout:** PurchaseOrderLayoutREST.I **Search Layout:** FindVendor_PurchaseOrder.I **Container:** find_vendor

Address Screens

The **To** and **From** search screens allow you to look up addresses you commonly use as well as any customer addresses from the **Mileage Sheet** screen.



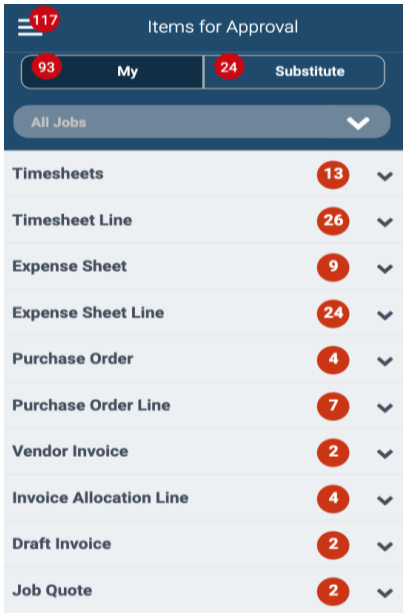


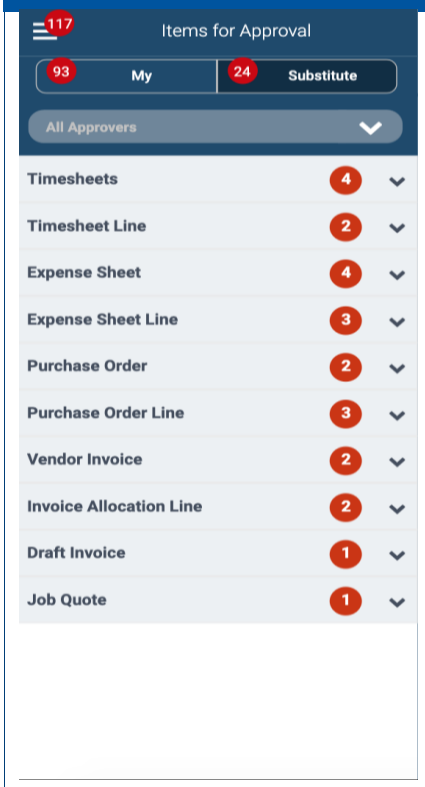
Navigation Path	REST
Mileage Sheet » Mileage Sheet » Mileage Sheet Line » From	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** FindJobHeader_Expense.I **Container:**

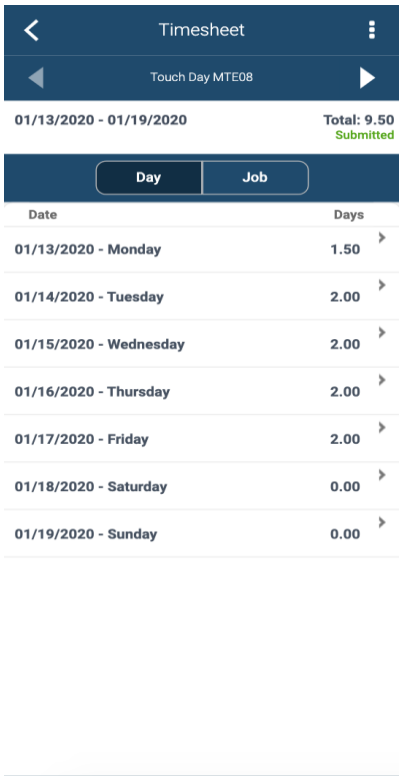
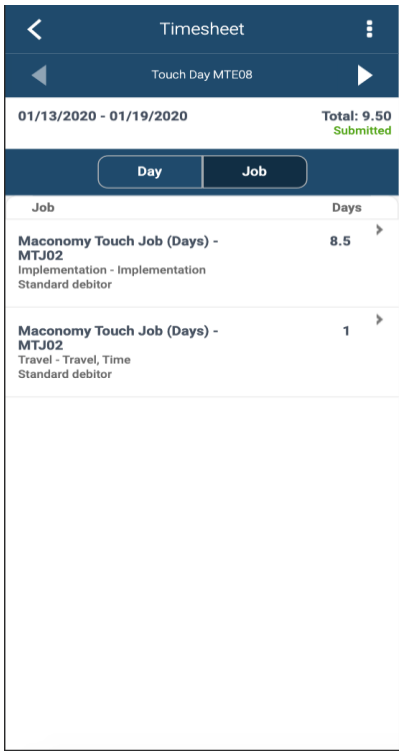
Navigation Path	REST
	Find_RecentlyUsedMileageFromU
Mileage Sheet » Mileage Sheet » Mileage Sheet Line » To	**Screen Layout:** MileageSheetLineLayoutREST.I **Search Layout:** Find_LocationFromTo.I **Container:** Find_RecentlyUsedMileageToU

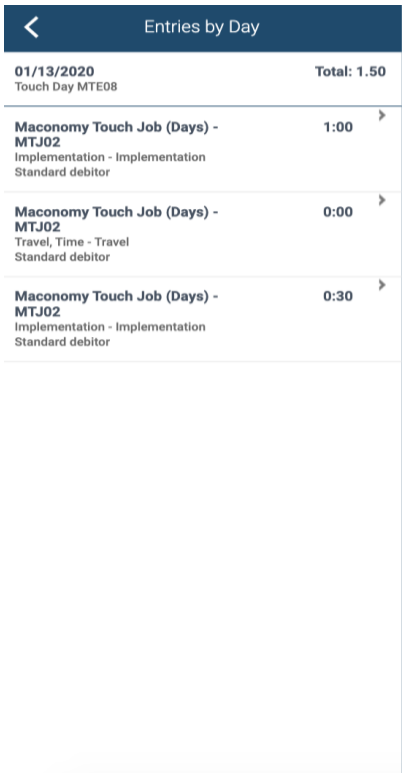
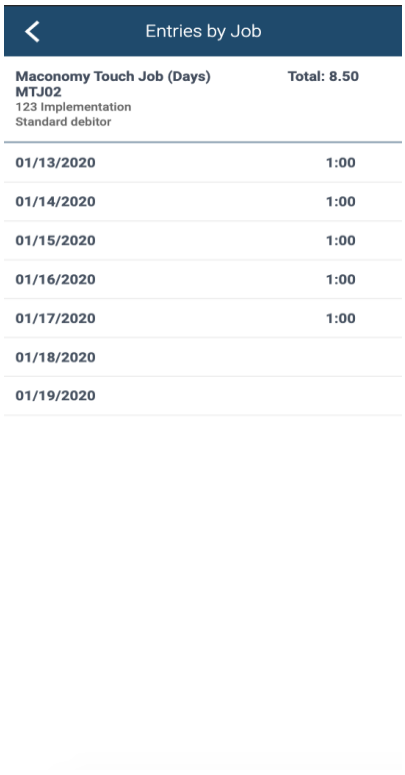
Approval Screens (REST based)

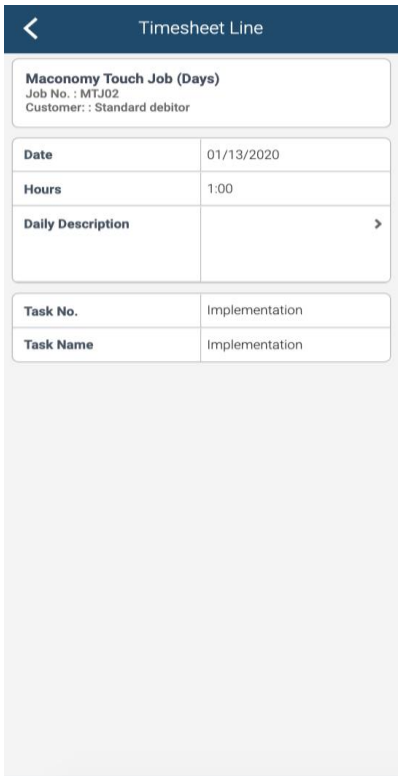
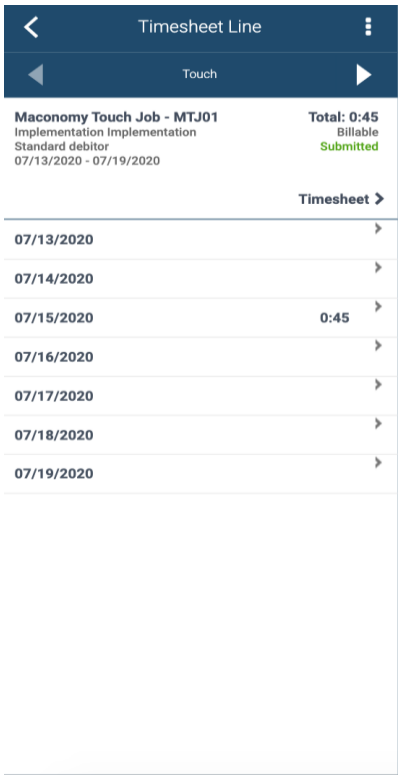
These screens are used if you have Maconomy 2.5.1 (or higher).

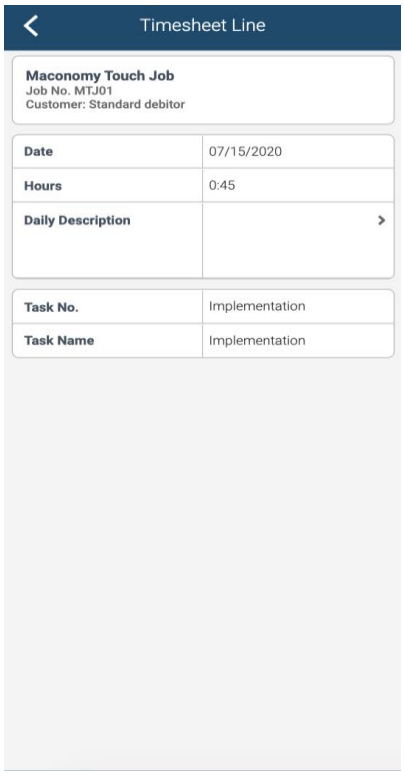
Screen	MScript	REST
	*Not applicable*	**Layout:** ItemsForApprovalRest.I **Containers:** Timesheets – approvetimesheetsbyemployee Timesheet Line – approvetimesheetlinesbyemployee Expense Sheets – approveexpensesheetsbyemployee Expense Sheet Line – approveexpensesheetlinesbyemployee Purchase Order – approvepurchaseordersbyemployee

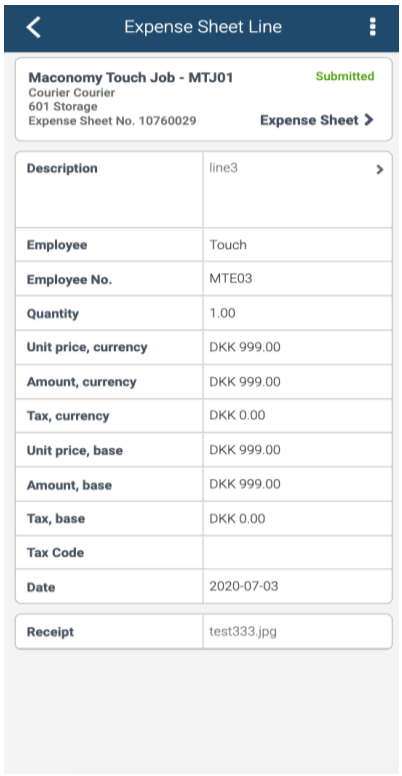
Screen	MScript	REST
		Purchase Order Line – approvepurchaseorderlinesbyemployee Vendor Invoice – approvevendorinvoicesbyemployee Invoice Allocation Line – approvevendorinvoicealloclinesbyemployee Draft Invoice – approvedraftinvoicesbyemployee Job Quote – approvejobquotesbyemployee Absence – absencecalendarlines Allowance – absenceentryoverview

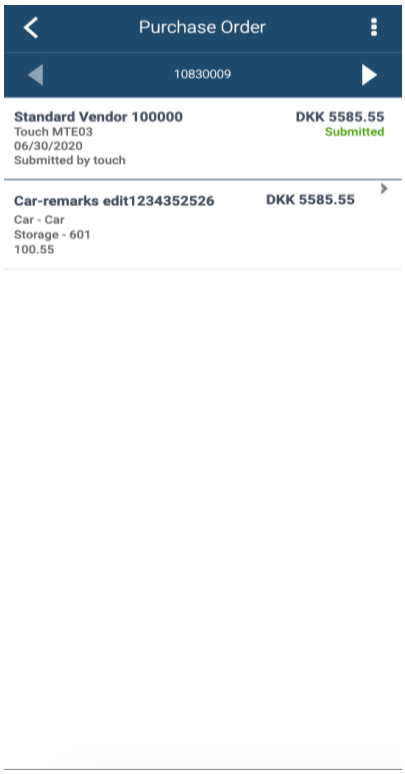
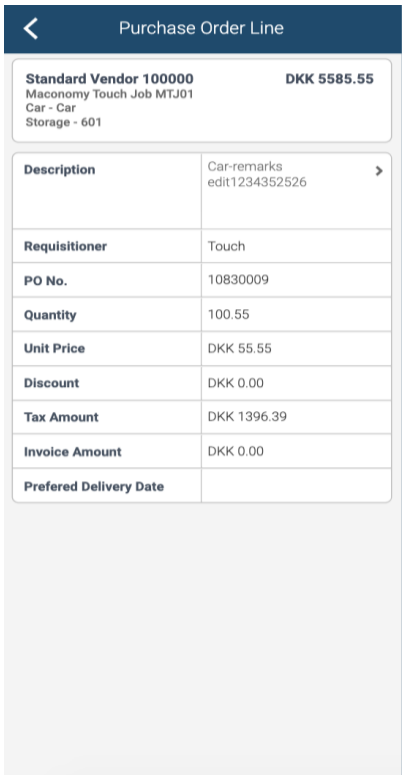
Screen	MScript	REST
 	*Not applicable*	**Layout:** ApprovalTimeSheetLayoutREST.l **Containers:** Day Tab – timesheets maconomylongtext:timesheets Job Tab – find_timesheetlinesummaryu

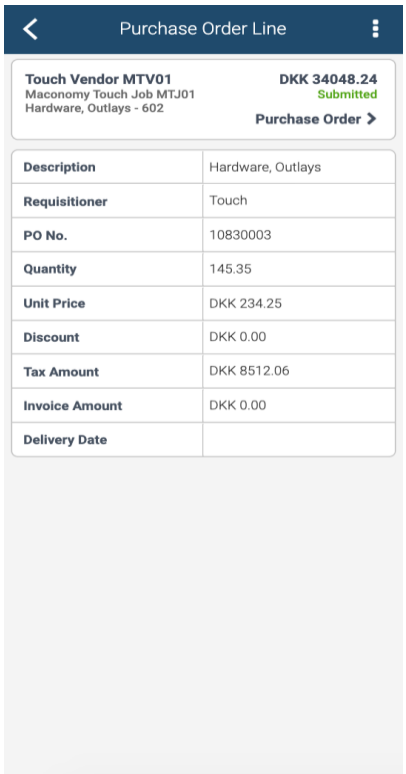
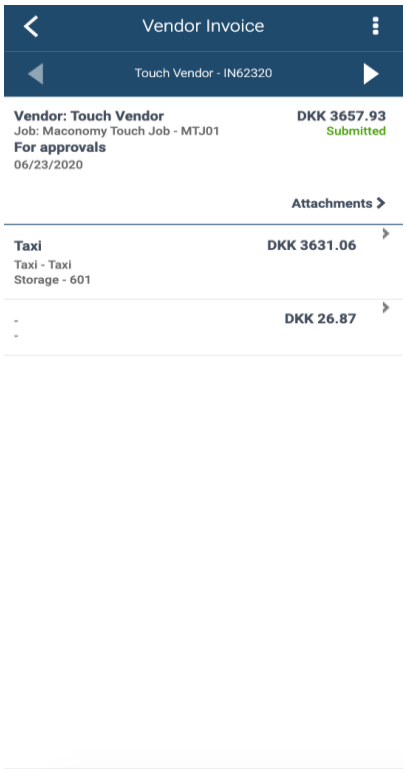
Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalTimesheetEntriesByDayLayoutREST.I **Container:** showpurchaseorderlines
	*Not applicable*	**Layout:** ApprovalTimesheetEntriesByJobLayoutREST.I **Container:** find_timesheetlinesummaryu

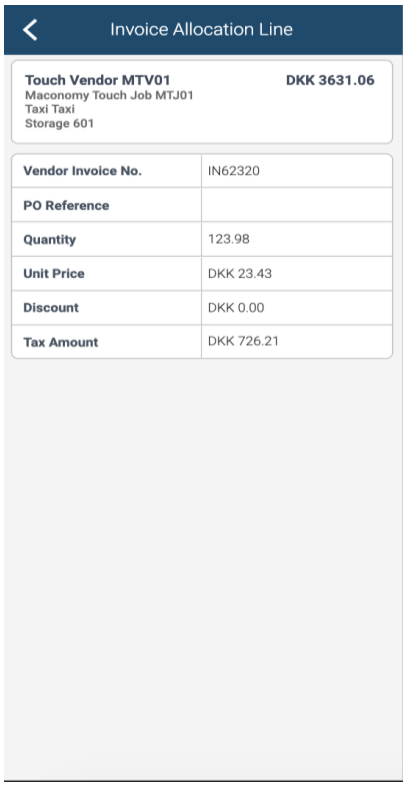
Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalTimesheetLineLayoutREST.I **Container:** timesheets maconomylongtext:timesheets
	*Not applicable*	**Layout:** PmTimesheetLineLayoutREST.I **Container:** timeregistrationdetails maconomylongtext:timeregistrationdetails

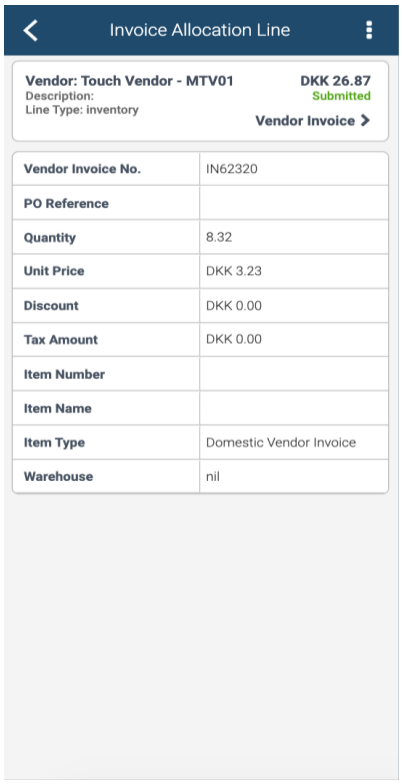
Screen	MScript	REST
	*Not applicable*	**Layout:** PmTimesheetLogEntryLayoutREST.I **Container:** timeregistrationdetails maconomylongtext:timeregistrationdetails
	Not applicable	**Layout:** ApprovalExpenseSheetLayoutREST.I **Containers:** expensesheets

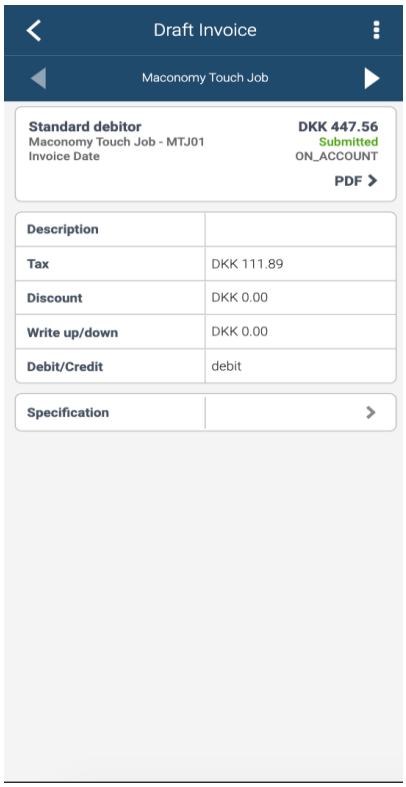
Screen	MScript	REST
	Not applicable	**Layout:** ApprovalExpenseSheetLineLayoutREST.I **Containers:** expensesheets
	*Not applicable*	**Layout:** ApprovalExpenseSheetLinePMLayoutREST.I **Containers:** expensesheetlines

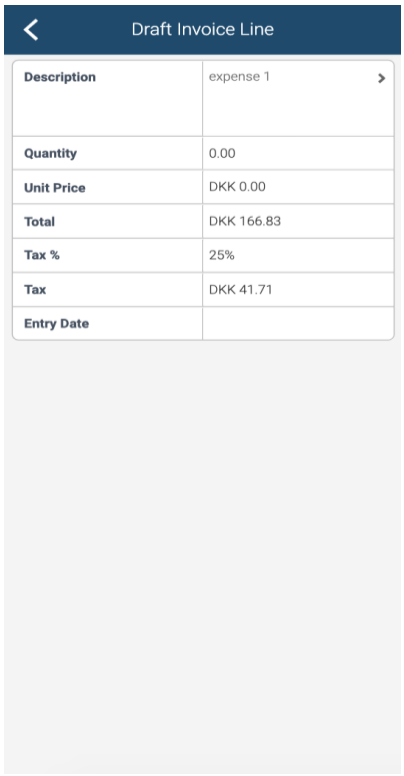
Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalPurchaseOrderLayoutREST.I **Containers:** purchaseorders maconomylongtext:purchaseorders
	*Not applicable*	**Layout:** ApprovalPurchaseOrderLinePMLayoutREST.I **Containers:** purchaseorders maconomylongtext:purchaseorders

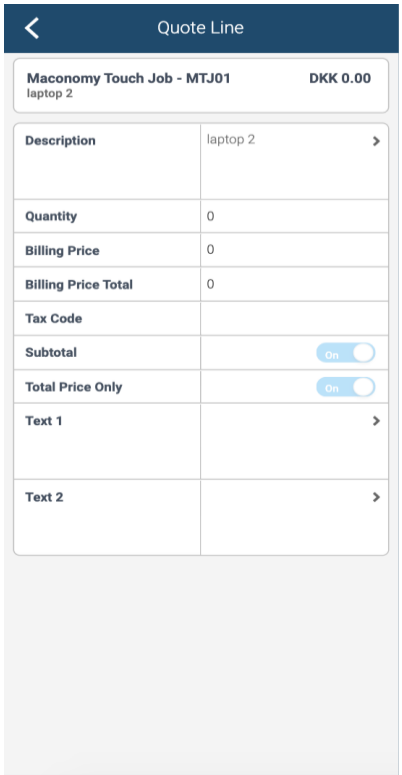
Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalPurchaseOrderLineLineLayoutREST.I **Container:** showpurchaseorderlines
	*Not applicable*	**Layout:** ApprovalVendorInvoiceLayoutREST.I **Container:** invoiceallocation maconomylongtext:invoiceallocation

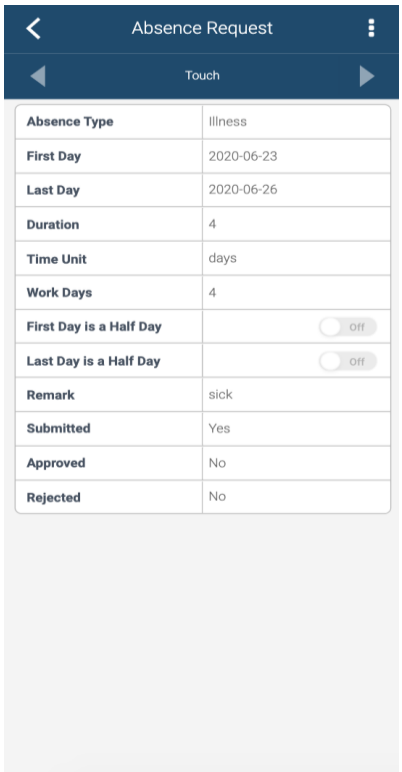
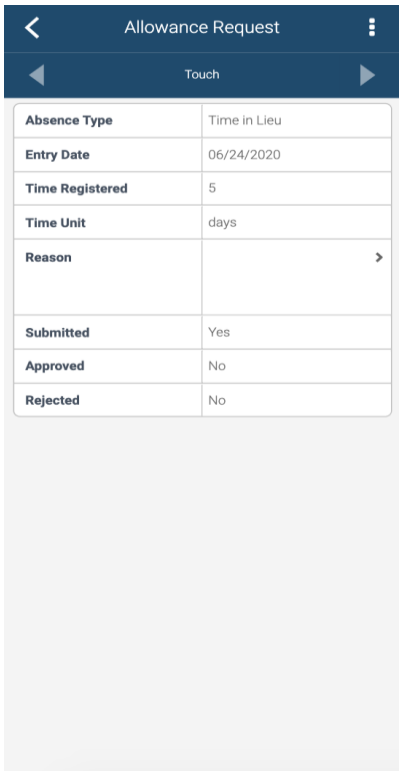
Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalVendorInvoiceAllocationLineLayoutREST.l **Container:** invoiceallocation maconomylongtext:invoiceallocation

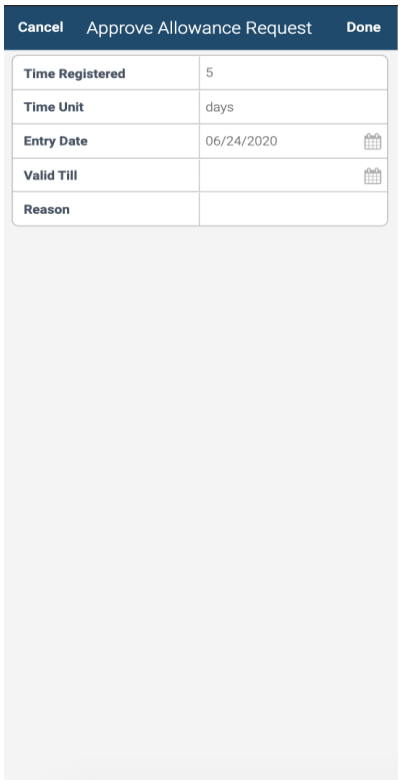
Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalVendorInvoiceAllocationLineLinePMLayoutREST.I **Container:** showinvoiceallocationlines

Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalInvoiceDraftLayoutREST.I **Container:** invoiceediting maconomylongtext:invoiceediting

Screen	MScript	REST
	Not applicable	**Layout:** ApprovalInvoiceDraftLinesLayoutREST **Container:** invoiceediting maconomylongtext:invoiceediting
	*Not applicable*	**Layout:** ApprovalInvoiceDraftLineLayoutREST.l **Container:** invoiceediting maconomylongtext:invoiceediting

Screen	MScript	REST
	Not applicable	**Layout:** ApprovalQuoteLineLayoutRESTI **Container:** quoteediting maconomylongtext:quoteediting
	*Not applicable*	**Layout:** ApprovalQuoteLineLayoutRESTI **Container:** quoteediting maconomylongtext:quoteediting

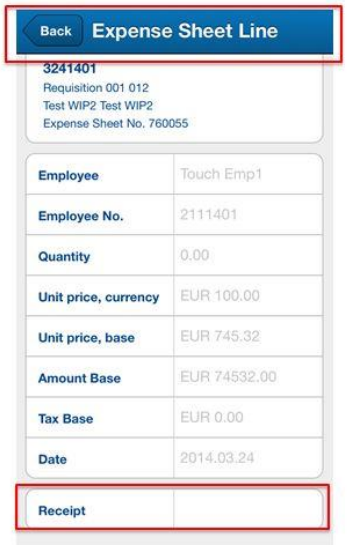
Screen	MScript	REST
	*Not applicable*	**Layout:** AbsenceApprovalFormLayoutREST.I **Container:** absencecalendarlines
	*Not applicable*	**Layout:** AllowanceApprovalFormLayoutREST.I **Container:** absenceentryoverview

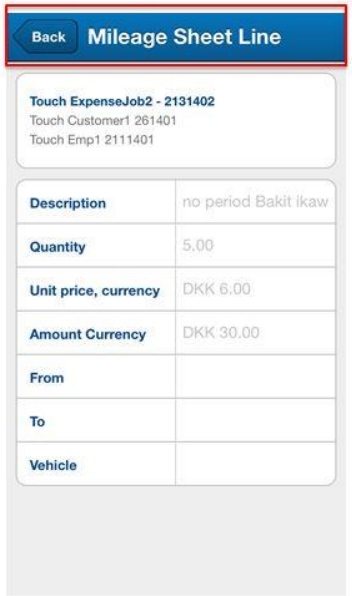
Screen	MScript	REST
	*Not applicable*	AllowanceApproveInputScreen REST.I

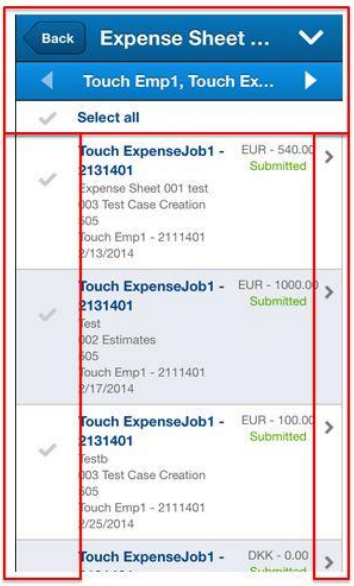
Approval Screens (MScript based)

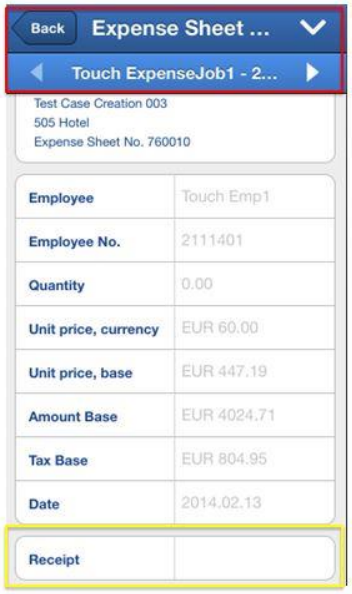
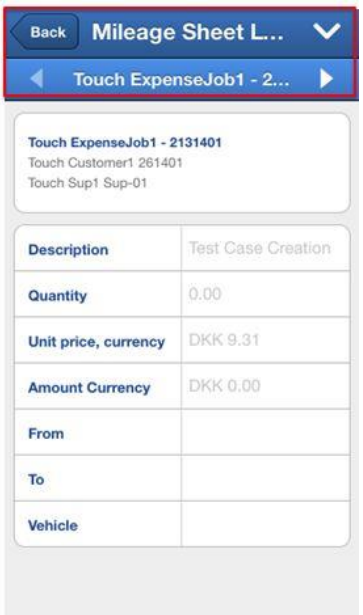
These screens are used if you have Maconomy 2.5 (or lower).

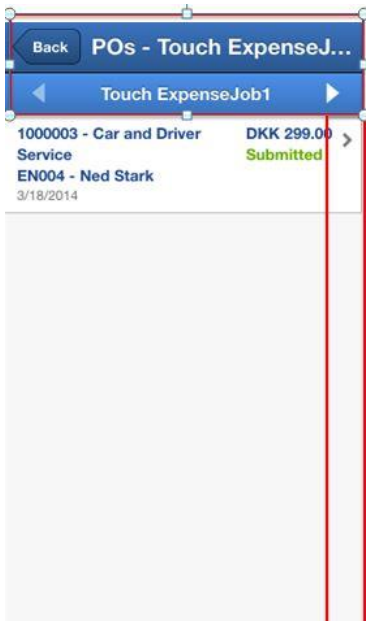
Note: The regions in the red boxes are not subject for customization. Deltek recommends that you do not customize the regions in the yellow boxes.

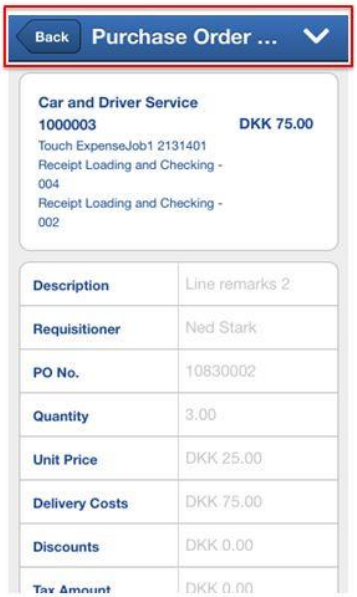
Screen	MScript	REST
	Layout: ApprovalExpenseSheetListLayout **Metadata:** ApprovalExpenseSheetListMetadata	ApprovalExpenseSheetLayoutREST.I
	**Layout:** ApprovalExpenseSheetLineLayout.I **Metadata:** ApprovalExpenseSheetLineMetadata.I	ApprovalExpenseSheetLineLayoutREST.I

Screen	MScript	REST
	**Layout:** ApprovalMileageSheetLineLayout.l **Metadata:** ApprovalMileageSheetLineMetadata.l	ApprovalMileageSheetLayoutREST.l
	Layout: ApprovalExpenseJobListLayout.l **Metadata:** ApprovalExpenseJobListMetadata.l	*Not applicable*

Screen	MScript	REST
	Layout: ApprovalExpenseSheetLineEmployeeListLayout.l **Metadata:** ApprovalExpenseSheetLineEmployeeListMetadata.l	*Not applicable*
	**Layout:** ApprovalExpenseSheetLineListLayout.l **Metadata:** ApprovalExpenseSheetLineListMetadata.l	ApprovalExpenseSheetListLayoutREST.l

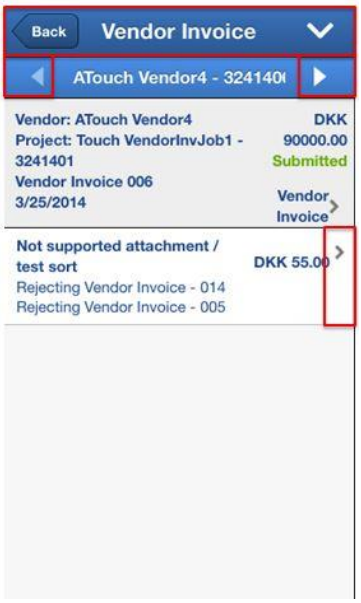
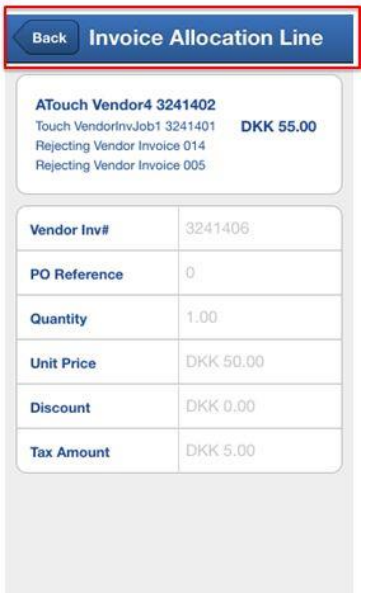
Screen	MScript	REST
	**Layout:** ApprovalExpenseSheetLineLayout.l **Metadata:** ApprovalExpenseSheetLineMetadata.l	ApprovalExpenseSheetLineLayoutREST.l
	**Layout:** ApprovalMileageSheetLineLayout.l **Metadata:** ApprovalMileageSheetLineMetadata.l	ApprovalMileageSheetLineLayoutREST.l

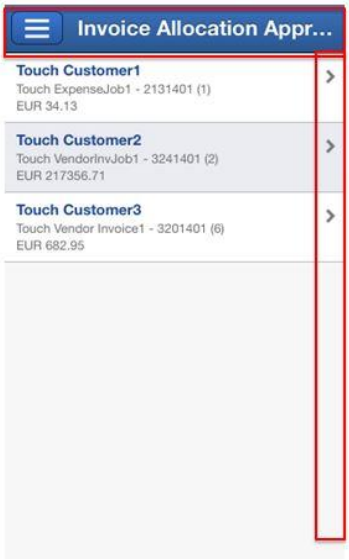
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderJobList Layout.l **Metadata:** ApprovalPurchaseOrderJobList Metadata.l	*Not applicable*
	**Layout:** ApprovalPurchaseOrderListLay out.l **Metadata:** ApprovalPurchaseOrderListMet adata.l	ApprovalPurchaseOrderListLay outREST.l

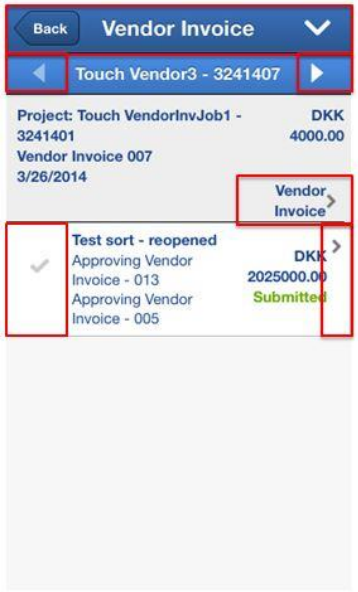
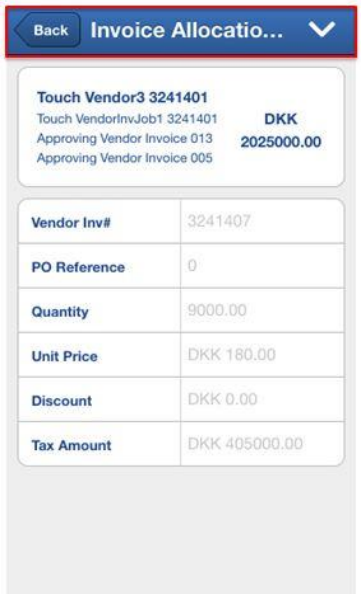
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderPMLayout.l **Metadata:** ApprovalPurchaseOrderPMMetadata.l	*Not applicable*
	**Layout:** ApprovalPurchaseOrderLinePMLayout.l **Metadata:** ApprovalPurchaseOrderLinePMMetadata.l	*Not applicable*

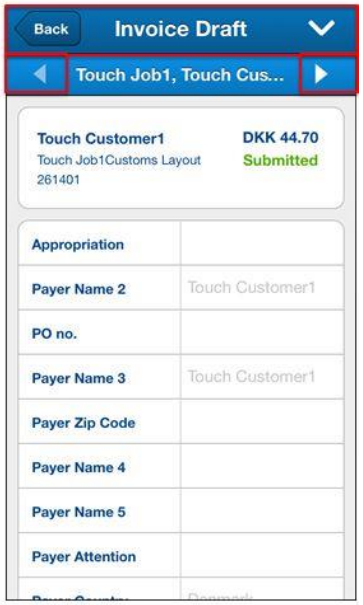
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderListLayout.l **Metadata:** ApprovalPurchaseOrderListMetadata.l	*Not applicable*
	Layout: ApprovalPurchaseOrderLayout.l **Metadata:** ApprovalPurchaseOrderMetadata.l	*Not applicable*

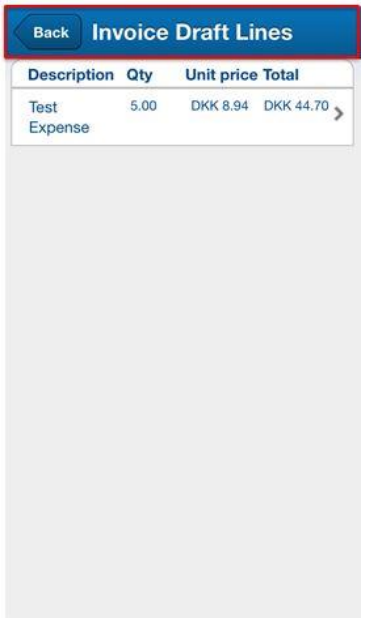
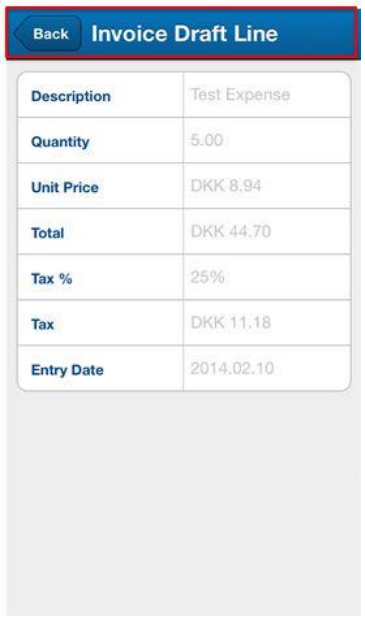
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderLineLayout.l **Metadata:** ApprovalPurchaseOrderLineMetadata.l	*Not applicable*
	Layout: ApprovalVendorInvoiceListLayout.l **Metadata:** ApprovalVendorInvoiceListMetadata.l	*Not applicable*

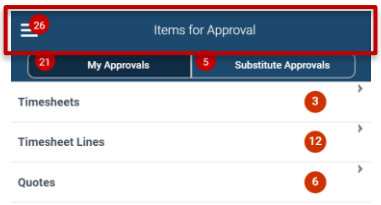
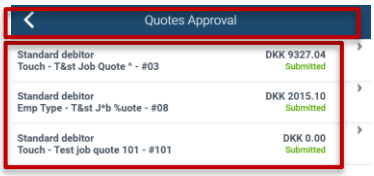
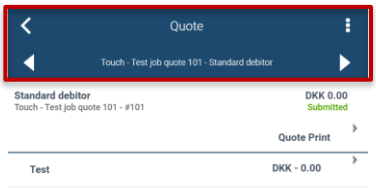
Screen	MScript	REST
	**Layout:** ApprovalVendorInvoiceLayout.l **Metadata:** ApprovalVendorInvoiceMetadata.l	*Not applicable*
	**Layout:** ApprovalVendorInvoiceAllocationLineLayout.l **Metadata:** ApprovalVendorInvoiceAllocationLineMetadata.l	*Not applicable*

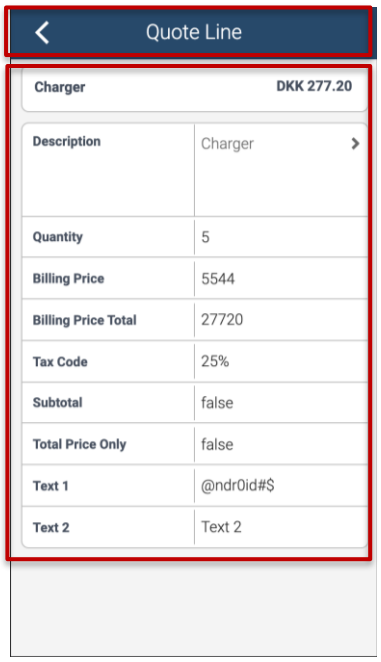
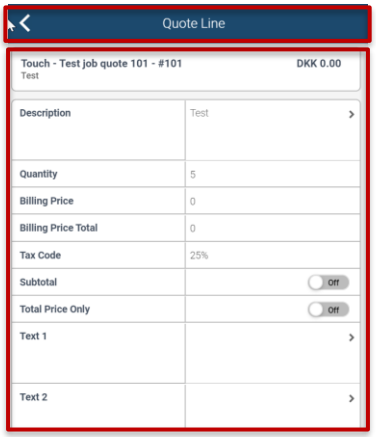
Screen	MScript	REST
	**Layout:** ApprovalVendorInvoiceAllocationLineJobListLayout.l **Metadata:** ApprovalVendorInvoiceAllocationLineJobListMetadata.l	*Not applicable*
	Layout: ApprovalVendorInvoiceListPMLayout.l **Metadata:** ApprovalVendorInvoiceListPMMetadata.l	*Not applicable*

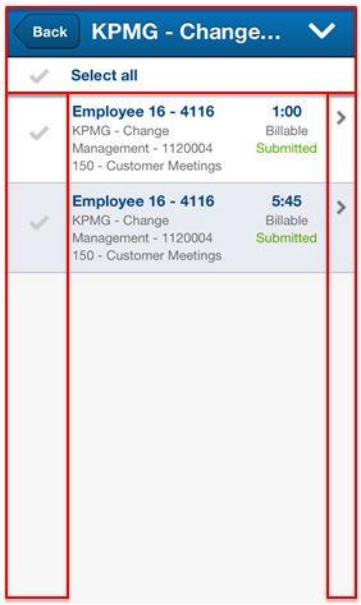
Screen	MScript	REST
	**Layout:** ApprovalVendorInvoicePMLayout.l **Metadata:** ApprovalVendorInvoicePMMetadata.l	*Not applicable*
	**Layout:** ApprovalVendorInvoiceAllocationLinePMLayout.l **Metadata:** ApprovalVendorInvoiceAllocationLinePMMetadata.l	*Not applicable*

Screen	MScript	REST
	Layout: ApprovalInvoiceDraftListLayout.l **Metadata:** ApprovalInvoiceDraftListMetadata.l	ApprovalInvoiceDraftListLayoutREST.l
	**Layout:** ApprovalInvoiceDraftLayout.l **Metadata:** ApprovalInvoiceDraftMetadata.l	ApprovalInvoiceDraftLayoutREST.l

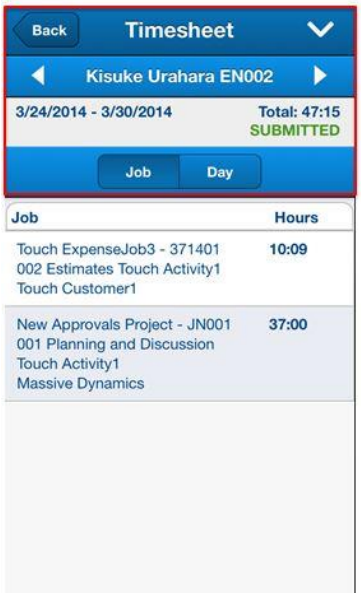
Screen	MScript	REST
	**Layout:** ApprovalInvoiceDraftLinesLayout.l **Metadata:** ApprovalInvoiceDraftLinesMetadata.l	*Not applicable*
	**Layout:** ApprovalInvoiceDraftLineLayout.l **Metadata:** ApprovalInvoiceDraftLineMetadata.l	*Not applicable*

Screen	MScript	REST
	*Not applicable*	**Layout:** ItemsForApprovalLayout.l
	*Not applicable*	**Layout:** ApprovalQuoteListLayoutREST.l **Container:** approvejobquotesbyemployee
	*Not applicable*	**Layout:** ApprovalQuoteLayoutREST.l **Container:** ▪ If **enableLongTextEmulation** is set to **false**: QuoteEditing ▪ If **enableLongTextEmulation** is set to **true**: maconomylongtext:QuoteEditing

Screen	MScript	REST
	*Not applicable*	**Layout:** ApprovalQuoteLineLayoutREST.l
	*Not applicable*	**Layout:** ApprovalQuoteLineLayoutREST.l **Container:** ▪ If **enableLongTextEmulation** is set to **false**: QuoteEditing ▪ If **enableLongTextEmulation** is set to **true**: maconomylongtext:QuoteEditing

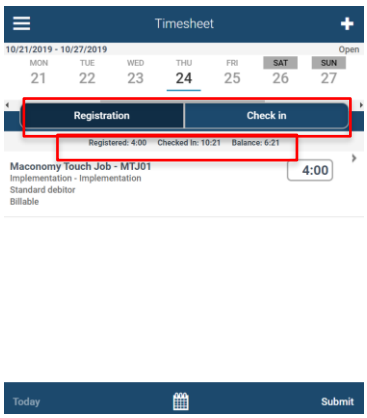
Screen	MScript	REST
	Layout: PmJobsLayout.l **Metadata:** PmJobsMetadata.l	*Not applicable*
	**Layout:** PmTimeSheetsLayout.l **Metadata:** PmTimeSheetsMetadata.l	*Not applicable*

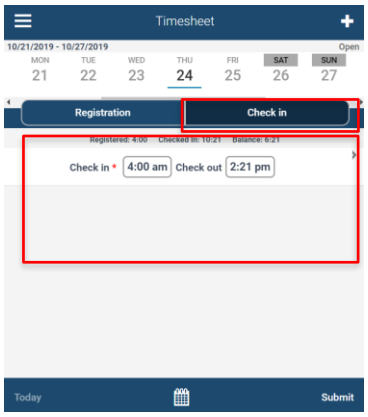
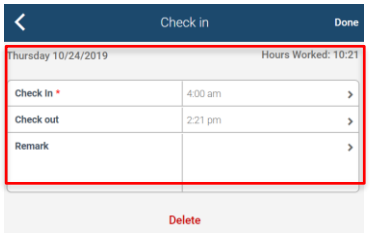
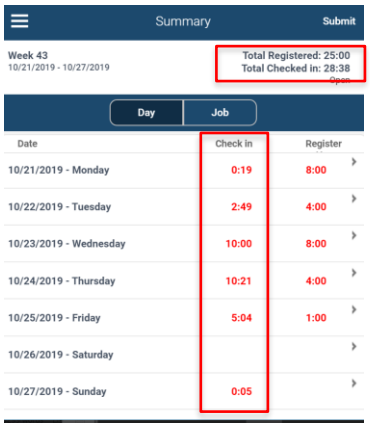
Screen	MScript	REST
	Layout: PmTimesheetLineLayout.l **Metadata:** PmTimesheetLineMetadata.l	*Not applicable*
	Layout: SupervisorApprovalListLayout.l **Metadata:** SupervisorApprovalListMetadata.l	*Not applicable*

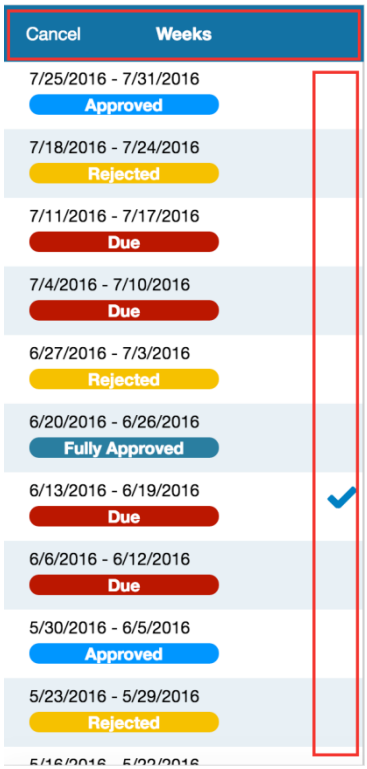
Screen	MScript	REST
	**Layout:** SupervisorTSJobLayout.l **Metadata:** SupervisorTSJobMetadata.l	*Not applicable*
	Layout: SupervisorTSDayLayout.l **Metadata:** SupervisorTSDayMetadata.l	*Not applicable*

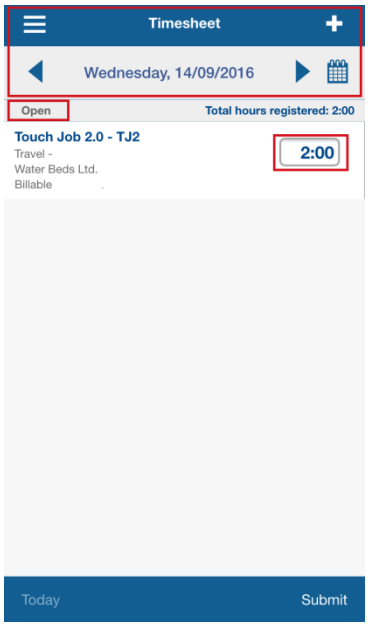
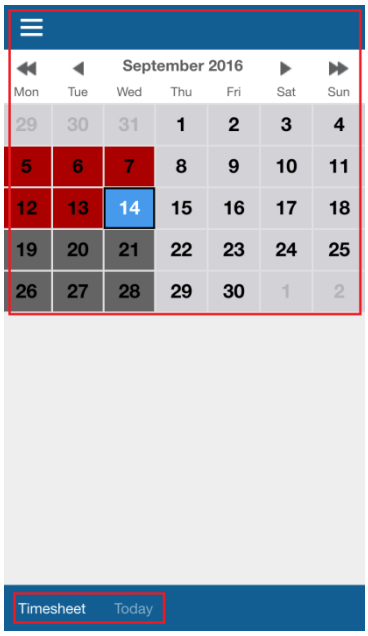
Screen	MScript	REST
	Layout: SupervisorEntriesByJobLayout.l **Metadata:** SupervisorEntriesByJobMetadata.l	*Not applicable*
	Layout: SupervisorEntriesByDayLayout.l **Metadata:** SupervisorEntriesByDayMetadata.l	*Not applicable*

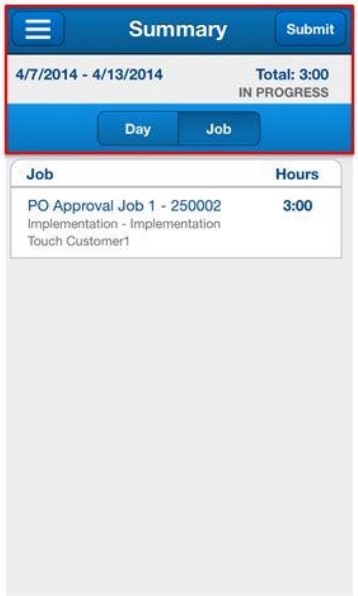
Time Registration Screens

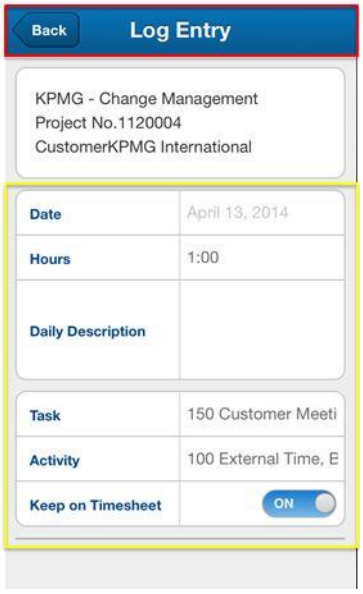
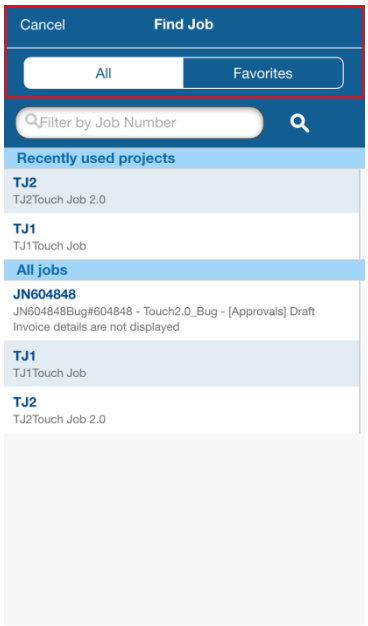
Screen	MScript	REST
	Layout: TimesheetLayout.I **Metadata:** TimesheetMetadata.I *Weekly Mode	**Layout:** TimesheetLayoutREST.I **Container:** TimeRegistration *Weekly Mode
	*Not applicable*	**Layout:** TimesheetLayoutREST.I **Container:** TimeRegistration If **CheckInEnabledDay<x>Var** is **true** (where <x> refers to day 1 to day 7), the Registration and Check in tabs display, and check In is required for a given day. If it is set to **false**, there are no changes to the UI. The Value: CheckInTimeForPeriodVar only applies if check in is required. If **CheckInBalanceForPeriodVar** is < or > 0, the total fields should be in red color.

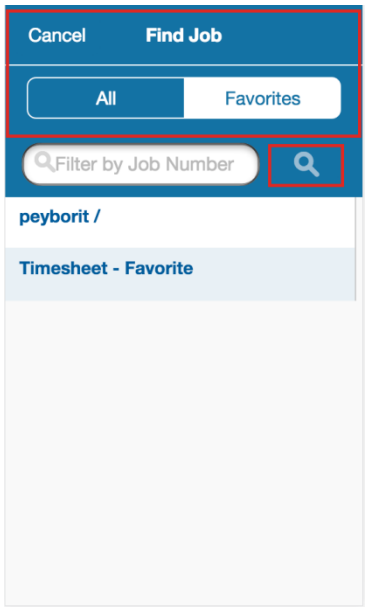
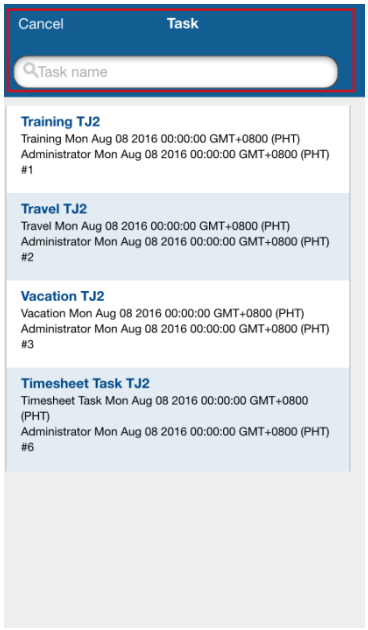
Screen	MScript	REST
	*Not applicable*	**Layout:** TimesheetLayoutREST.I (for Weekly Mode) TimesheetDailyLayoutREST (for Daily Mode) **Container:** DailyCheckIn To get from the TimeRegistration container to the DailyCheckIn, container you need to create the URL. Starting Touch 3.4 API, the following are no longer customizable: ▪ Registered: x ▪ Checked in: y Balance: z
	*Not applicable*	**Layout:** CheckInLayoutREST.I **Container:** DailyCheckIn
	*Not applicable*	**Layout:** SummaryLayoutREST.I (for Weekly) SummaryDailyLayoutREST.I (for Daily Mode) **Container:** TimeRegistration This only applies if **CheckInEnabledForPeriodVar** is set to **true**.

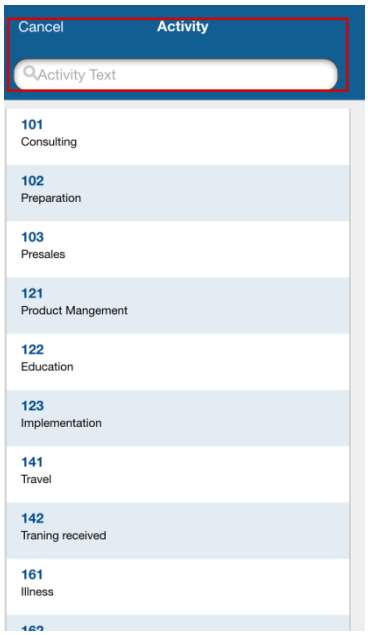
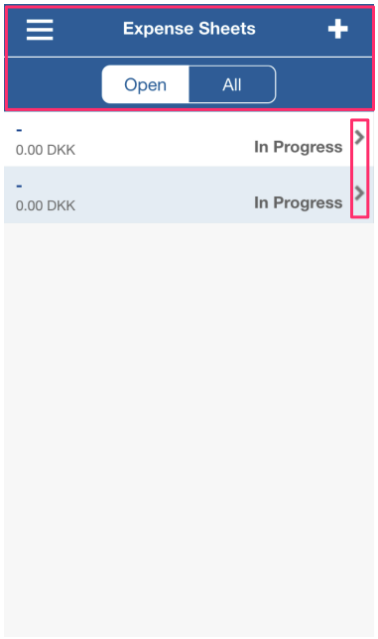
Screen	MScript	REST
		Total Checked In: Value: CheckInTimeForPeriodVar If **CheckInBalanceForPeriodVar** is < or > 0, the total fields should be in red color. **Checked In** column: Value: CheckInTimeRoundedDay<X>var The value only displays if **CheckInEnabledDay<x>Var** is set to **true**. **If CheckInBalanceDay<x>Var** is < or > 0, the values should be in red color.
	*Not applicable*	**Layout:** TimeSheetWeekLayoutREST.l

Screen	MScript	REST
	**Layout:** TimesheetLayout.l **Metadata:** TimesheetMetadata.l * Daily Mode	**Layout:** TimesheetDaysLayoutREST.l **Container:** DailyTimeRegistration * Daily Mode
	*Not applicable*	**Layout:** TimeSheetCalendarLayoutRESTT.l *Only the colors for each status are customizable.* **Container:** DailyTimeRegistration

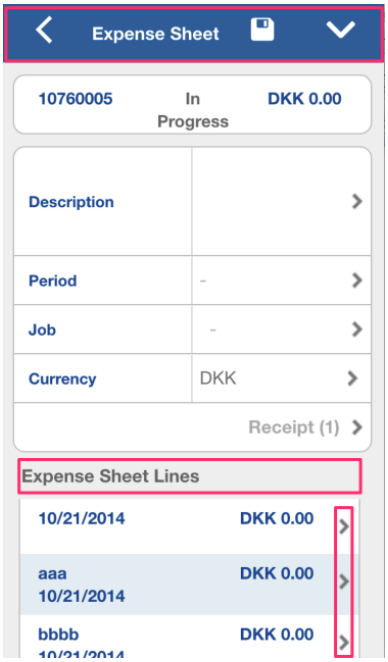
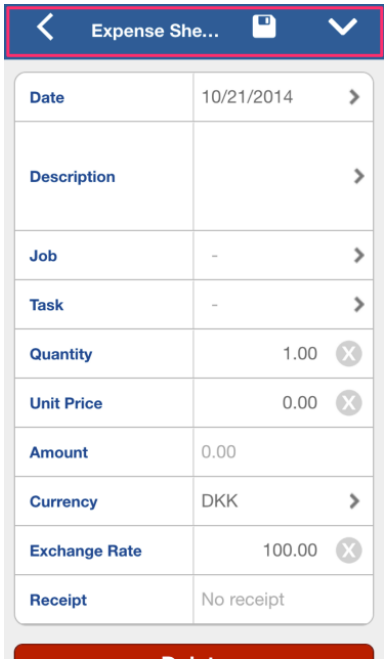
Screen	MScript	REST
  	**Layout:** SummaryLayout.I **Metadata:** SummaryMetadata.I	**Layout:** Weekly Mode – SummaryLayoutREST.I Daily Mode – SummaryDailyLayoutREST.I **Container:** Weekly Mode – TimeRegistration Find_TimeSheetLineSummaryU Daily Mode – DailyTimeRegistration Find_TimeSheetLineSummaryU

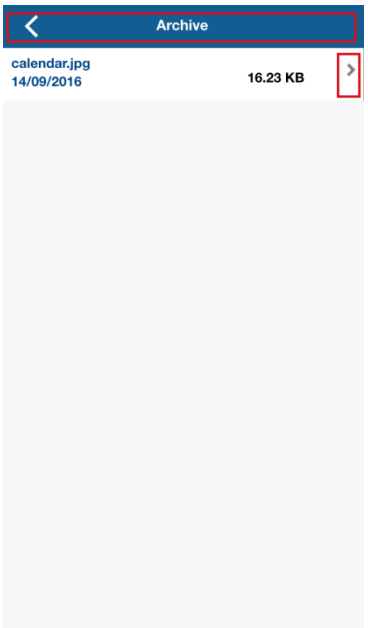
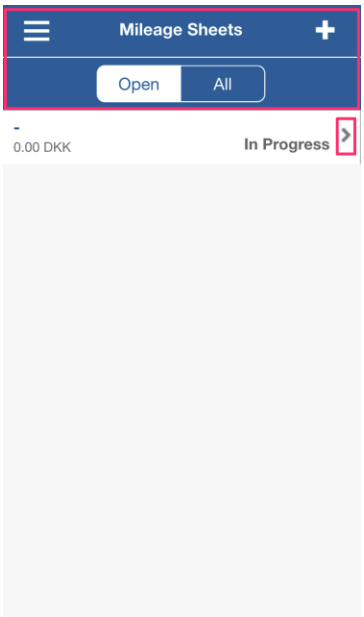
Screen	MScript	REST
	**Layout:** TimesheetLogentryLayout.I, DeltekTouch.I ▪ The new layout framework does not apply to the region in yellow box. It uses the old solution, which uses DeltekTouch.I. ▪ Three screens uses the same layout: Project Manager Timesheet Approval, Supervisor Timesheet Approval, and Timesheet **Metadata:** TimesheetLogEntryMetadata.I.	**Layout:** Weekly Mode – TimesheetLogEntryWeeklyLayoutREST.I Daily Mode – TimesheetLogEntryDailyLayoutREST.I **Container:** Weekly Mode – TimeRegistration Daily Mode – DailyTimeRegistration
	*Not applicable*	**Layout:** Find_JobHeader_Timesheet.I **Container:** Find_TimeSheetLineU Find_JobEmployeeControlU

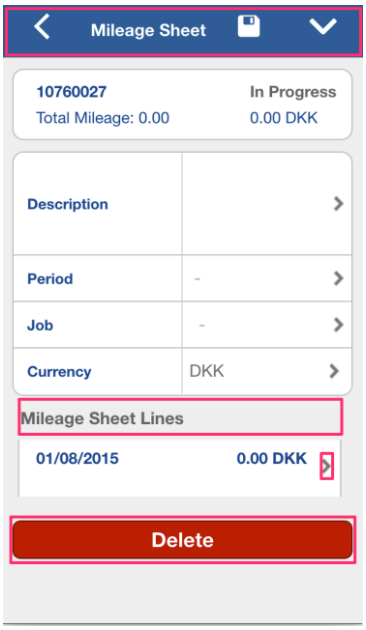
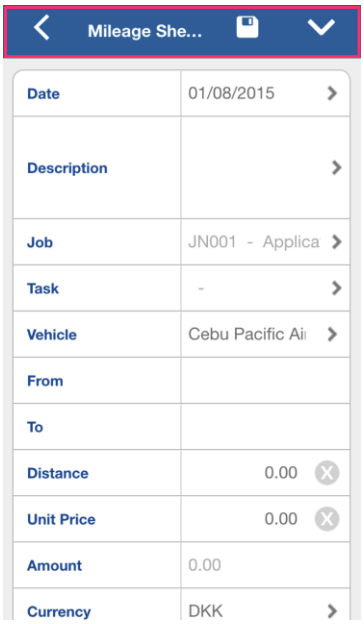
Screen	MScript	REST
	*Not applicable*	**Layout:** Find_JobFavorite_Timesheet.l **Container:** Find_JobFavorite
	*Not applicable*	**Layout:** Find_TaskListLine.l **Container:** Find_TaskListLine

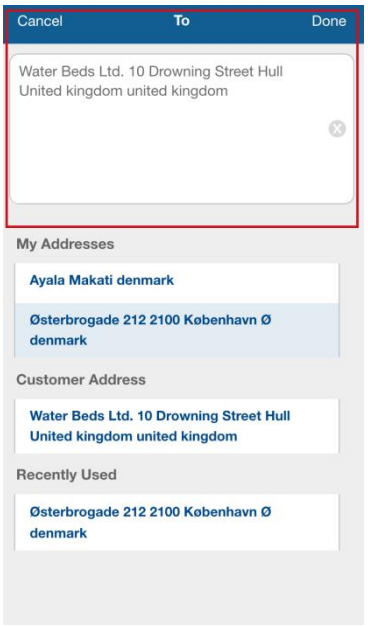
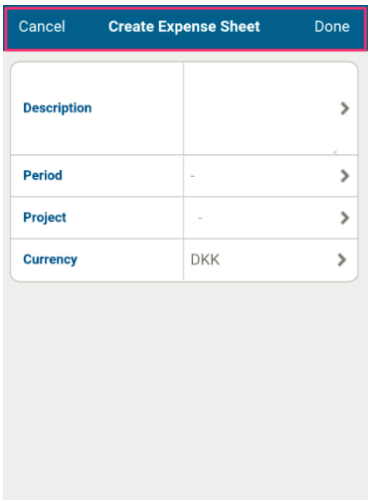
Screen	MScript	REST
	*Not applicable*	**Layout:** Find_Activity.l **Container:** Find_Activity
	**Layout:** ExpenseSheetsLayout.l **Metadata:** ExpenseSheetsMetadata.l	**Layout:** ExpenseSheetsLayoutREST.l **Container:** ExpenseSheets

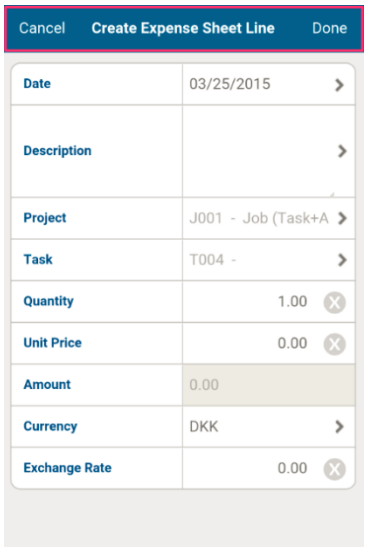
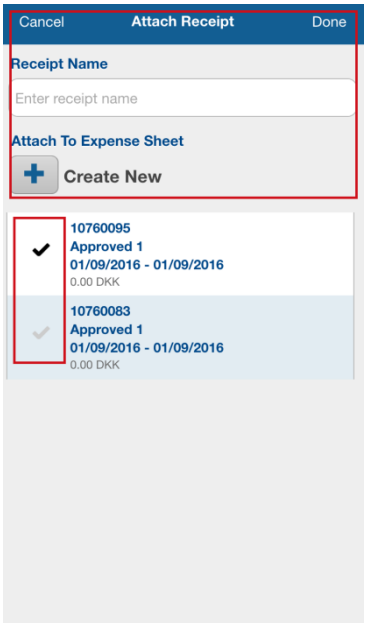
Expense and Mileage Registration Screens

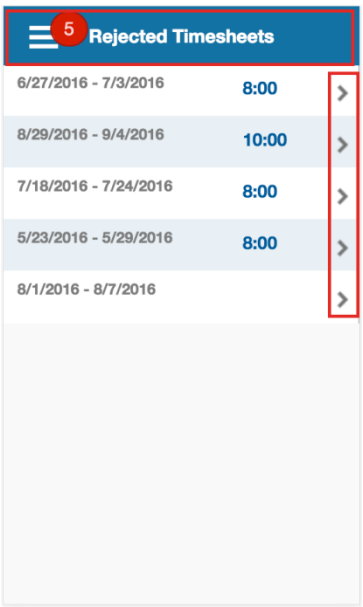
Screen	MScript	REST
	**Layout:** ExpenseSheetLayout.I **Metadata:** ExpenseSheetMetadata.I	**Layout:** ExpenseSheetLayoutREST.I **Container:** ExpenseSheets
	**Layout:** ExpenseSheetLineLayout.I **Metadata:** ExpenseSheetLineMetadata.I	**Layout:** ExpenseSheetLineLayoutREST.I **Container:** ExpenseSheets

Screen	MScript	REST
	*Not applicable*	**Layout:** DocumentArchiveListREST.l **Container:** DocumentArchives
	**Layout:** MileageSheetsLayout.l **Metadata:** MileageSheetsMetadata.l	**Layout:** MileageSheetsLayoutREST.l **Container:** MileageSheets

Screen	MScript	REST
	**Layout:** MileageSheetLayout.I **Metadata:** MileageSheetMetadata.I	**Layout:** MileageSheetLayoutREST.I **Container:** MileageSheets
	**Layout:** MileageSheetLineLayout.I **Metadata:** MileageSheetLineMetadata.I	**Layout:** MileageSheetLineLayoutREST.I **Container:** MileageSheets

Screen	MScript	REST
	*Not applicable*	**Layout:** Find_LocationFromTo.l **Container:** Find_RecentlyUsedMileageFromTou
	**Layout:** CreateExpenseSheetLayout.l **Metadata:** CreateExpenseSheetMetadata.l	*Not applicable*

Screen	MScript	REST
	**Layout:** CreateExpenseSheetLineLayout.l **Metadata:** CreateExpenseSheetLineMetadata.l	**Layout:** QuickCaptureExpenseSheetLineLayoutREST.l **Container:** ExpenseSheets
	*Not applicable*	**Layout:** QuickCaptureAttachReceiptLayoutREST.l **Container:** ExpenseSheets

Screen	MScript	REST
	**Layout:** RejectedTimesheetsLayout.l **Metadata:** RejectedTimesheetsMetadata.l	**Layout:** RejectedTimesheetsLayoutREST.l **Container:** Find_RejectedTimeSheet U

Appendix B: If You Need Assistance

If you need assistance installing, implementing, or using Deltek Touch for Maconomy, Deltek makes a wealth of information and expertise readily available to you.

Customer Services

For over 30 years, Deltek has maintained close relationships with client firms, helping with their problems, listening to their needs, and getting to know their individual business environments. A full range of customer services has grown out of this close contact, including the following:

- Extensive self-support options through the Deltek Support Center.
- Phone and email support from Customer Care analysts
- Technical services
- Consulting services
- Custom programming
- Classroom, on-site, and Web-based training

Attention: Find out more about these and other services from the Deltek Support Center.

Deltek Support Center

The Deltek Support Center is a support Web site for Deltek customers who purchase an Ongoing Support Plan (OSP).

The following are some of the many options that the Deltek Support Center provides:

- Search for product documentation, such as release notes, install guides, technical information, online help topics, and white papers
- Ask questions, exchange ideas, and share knowledge with other Deltek customers through the Deltek Support Center Community
- Access Cloud-specific documents and forums
- Download the latest versions of your Deltek products
- Search Deltek's knowledge base
- Submit a support case and check on its progress
- Transfer requested files to a Customer Care analyst
- Subscribe to Deltek communications about your products and services
- Receive alerts of new Deltek releases and hot fixes
- Initiate a Chat to submit a question to a Customer Care analyst online

Attention: For more information regarding Deltek Support Center, refer to the online help available from the website.

Access Deltek Support Center

To access the Deltek Support Center:

1. Go to https://deltek.custhelp.com.
2. Enter your Deltek Support Center **Username** and **Password**.
3. Click **Login**.

Note: If you forget your username or password, you can click the **Need Help?** button on the login screen for help.

Additional Documentation

The following table lists the Deltek documentation available for this release. Except where noted, all the user guides and quick reference guides listed in this table are available for download from the Deltek Support Center.

Document Name	Description
Deltek Touch for Maconomy Installation Guide	This document provides instructions for the installation and configuration of application.
Deltek Touch for Maconomy Release Notes	This document contains a summary of the technical considerations, major features, and known issues of the application.
Deltek Touch for Maconomy User Guide	This document contains detailed information and instructions on how to use various features of the application.
Deltek Touch for Maconomy Upgrade Guide	*This document is only intended for Technical Consultants.* It describes necessary steps to upgrade to the latest Touch system.
Deltek Touch for Maconomy MScript and RESTful Web Services Comparison Guide	This document describes the differences between MScript and Representational State Transfer (REST) Web services to help technical consultants prepare for implementing Deltek Touch for new users or migrate their current Touch installation to REST.
Deltek FPU Release Notes for Maconomy Touch	*These documents are only intended for Technical Consultants.* There are four FPU release notes, each of which applies to the supported Maconomy core version. Each document contains a summary of the pre-installation information, enhancements, and API changes to Touch for each supported Maconomy version. It is part of the Deltek Touch for Maconomy product information suite and

Document Name	Description
	should only distributed internally within the Maconomy organization and to partners.

About Deltek

Better software means better projects. Deltek is the leading global provider of enterprise software and information solutions for project-based businesses. More than 23,000 organizations and millions of users in over 80 countries around the world rely on Deltek for superior levels of project intelligence, management and collaboration. Our industry-focused expertise powers project success by helping firms achieve performance that maximizes productivity and revenue. www.deltek.com