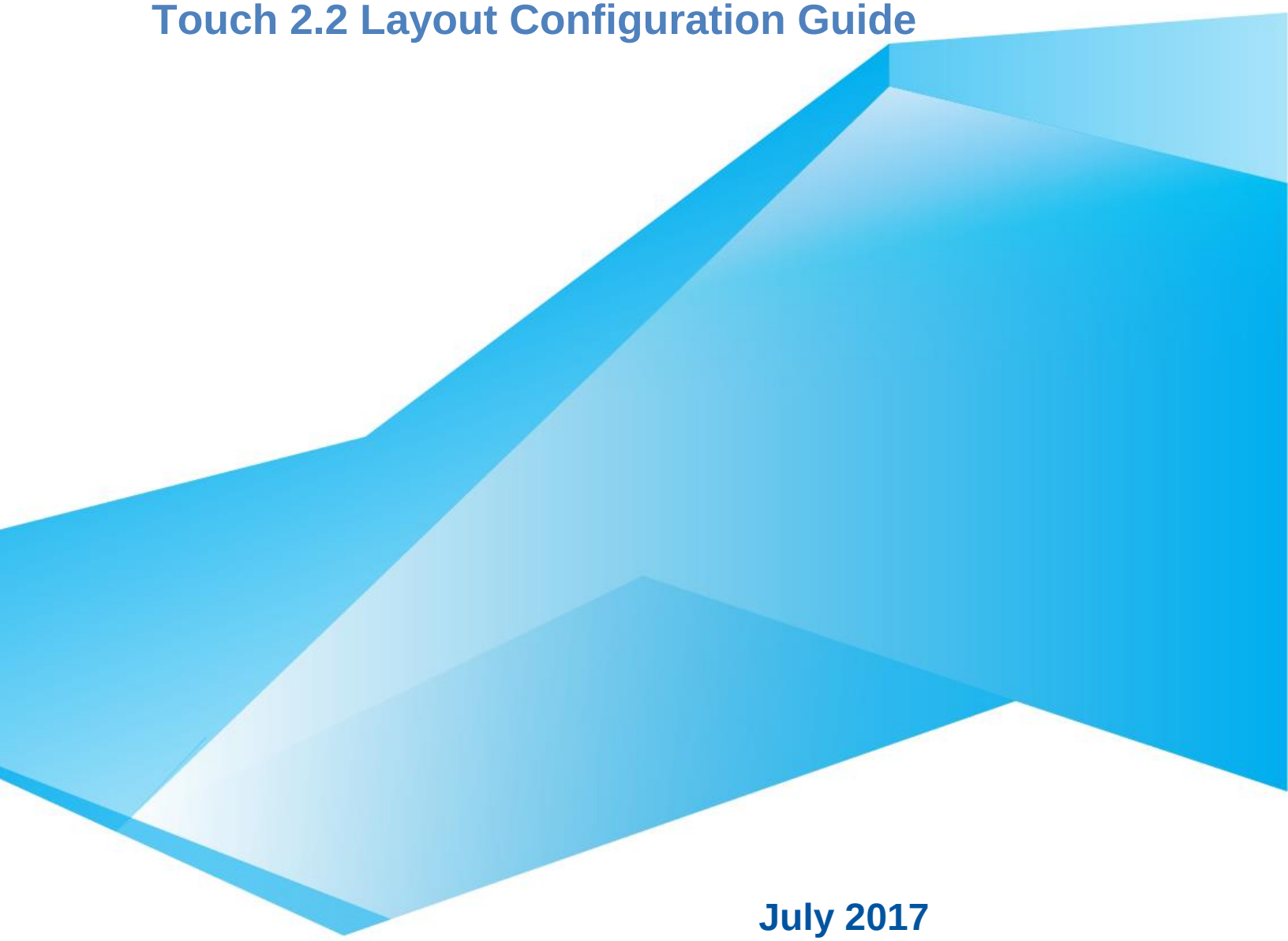
Deltek Know more. Do more.™

Deltek Touch for Maconomy

Touch 2.2 Layout Configuration Guide



July 2017

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 July 2017.

© 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 2
Configure the Timesheet Line or Add Job Screen 5
Configure an Approval or a Summary Screen 5
Adding New Search or Lookup field 6
Heavyweight Configuration 7
Query Configuration 8
Add Fields 11
Change Order 11
Change Condition 12
Layout Files 13
MScript Layout Files 13
REST Layout Files 13
Layout List Files 14
Layout File Table 16
Layout Configuration 21
Customization Capabilities 22
Layout Components and Attributes 24
Non-Editable Layout Components 24
Editable Layouts Components 24
Component Layout Attributes 25
Style Layout Attributes 29
Syntax Descriptions 36
Label 36
Field 37
Date Period Field 42
Project Lookup Field 44
Lookup Field 45
SearchField 47
LinearText 48
Grid 50
TextField 59

DocumentField 61
Group 63
Description 63
Rules 63
Example 63
Form 65
Attachment Icon 67
Location Picker 68
ReadOnlyField 69
Metadata Files 71
Appendix A: Screens, Layouts, and Metadata Files 73
Appendix B: For Additional Information 102
Customer Care Connect Site 102
Available Documentation for this Release 102

Overview

The Deltek Touch for Maconomy Layout Configuration Guide describes tasks necessary for the customization of layouts of the Deltek Touch for Maconomy 2.2 application using either MScript or RESTful Web services. For example, the customers may want to add or remove fields to/from a screen.

For technical information, such as pre-installation requirements and compatibility mode definition, see *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.

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*, *Google Play Store*, and *Windows 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.

Once you have configured the system, you must log on again to Deltek Touch to ensure that all changes take effect.

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 and Query Configuration** – *This item only applies to MScript.* This allows changes to queries used for search screens (such as Task List and Find Job) and 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.

For additional details please 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.

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

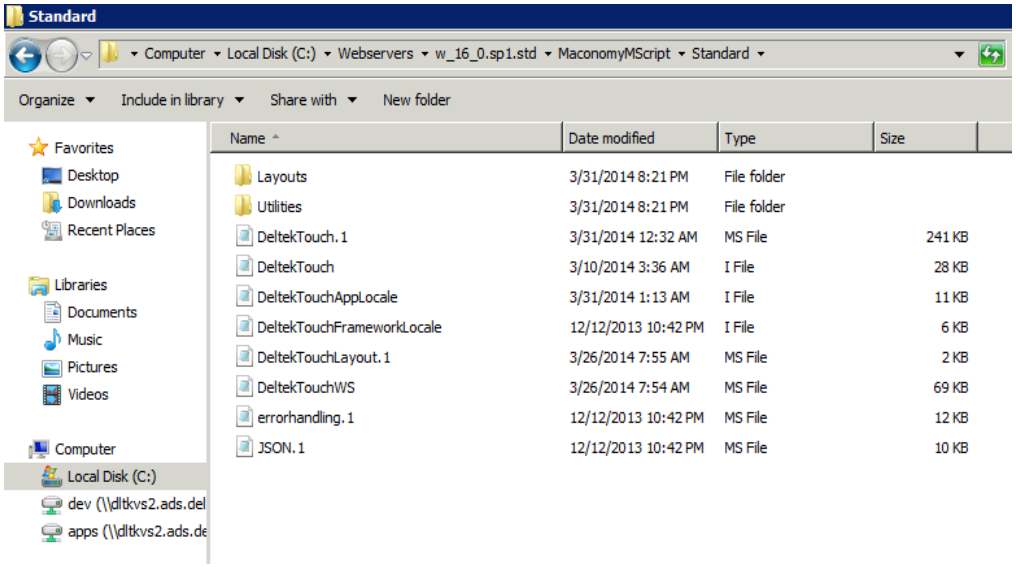
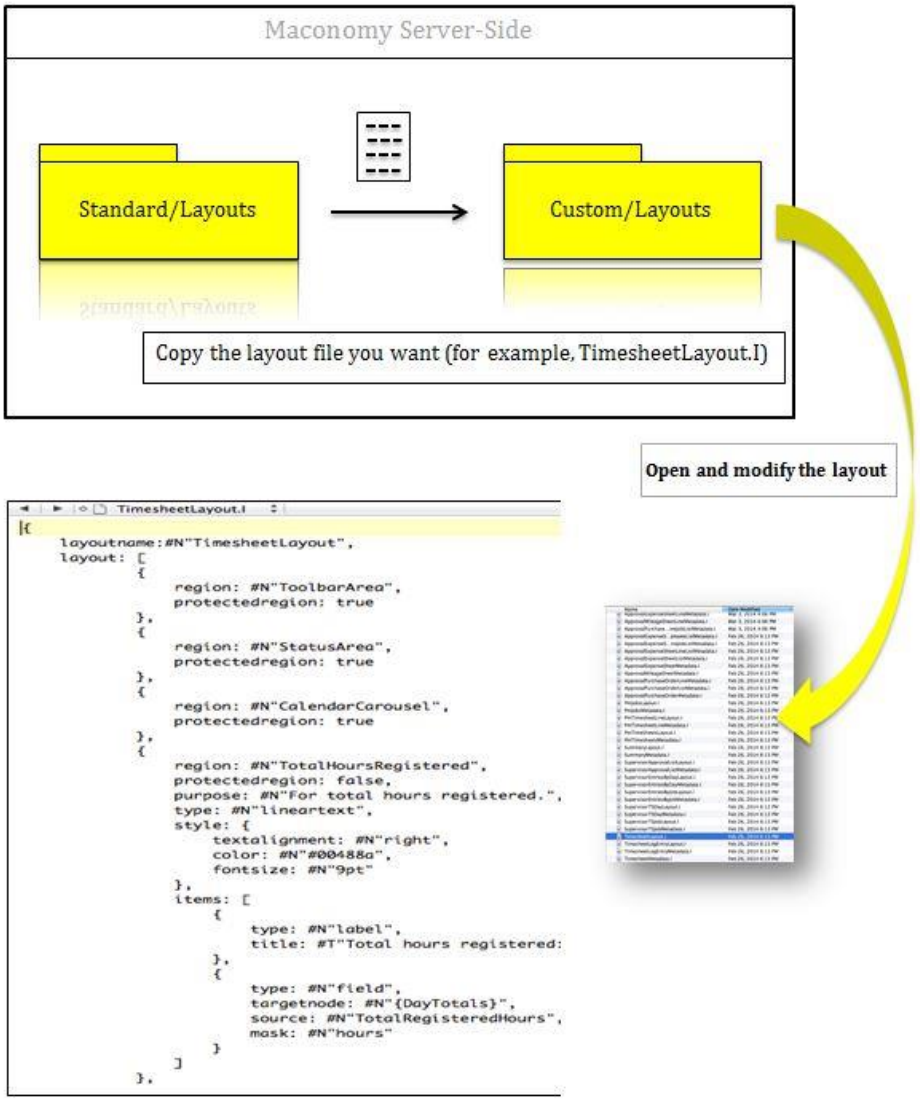
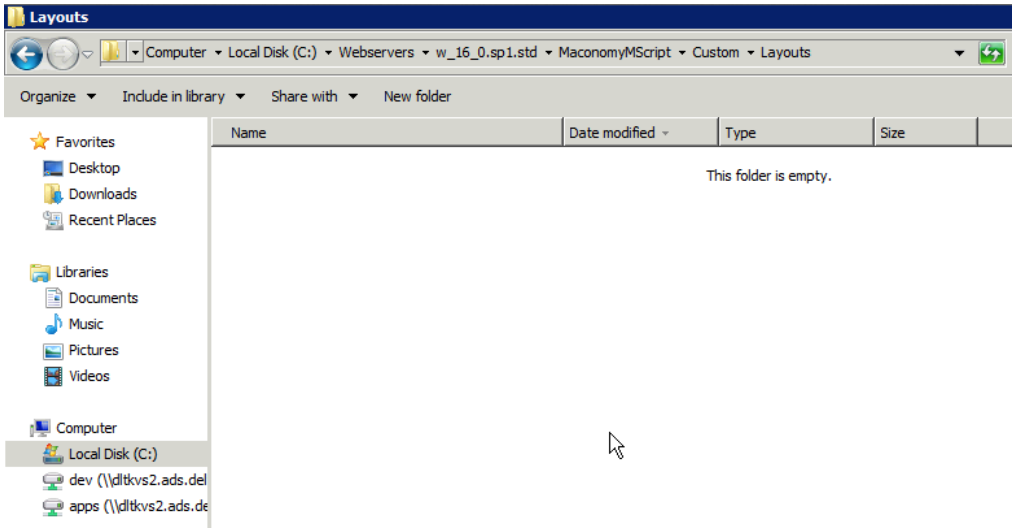

Standard
Computer
Local Disk (C:)
Webservers
w_16_0.sp1.std
MaconomyMScript
Standard
Organize
Include in library
Share with
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
dev (\\dltkvs2.ads.del
apps (\\dltkvs2.ads.de
Name
Date modified
Type
Size
Layouts 3/31/2014 8:21 PM File folder
Utilities 3/31/2014 8:21 PM File folder
DeltekTouch.1 3/31/2014 12:32 AM MS File 241 KB
DeltekTouch 3/10/2014 3:36 AM I File 28 KB
DeltekTouchAppLocale 3/31/2014 1:13 AM I File 11 KB
DeltekTouchFrameworkLocale 12/12/2013 10:42 PM I File 6 KB
DeltekTouchLayout.1 3/26/2014 7:55 AM MS File 2 KB
DeltekTouchWS 3/26/2014 7:54 AM MS File 69 KB
errorhandling.1 12/12/2013 10:42 PM MS File 12 KB
JSON.1 12/12/2013 10:42 PM MS File 10 KB

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.

> 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.

> 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.l** from **Standard/Layouts** to **Custom/Layouts**.
4. In the **Custom** folder, edit **TimesheetLogEntryLayout.l** 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

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

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.

Adding New Search or Lookup field

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

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"KeyValue",
              displayfields: [
              {
                source: #N"Specification3Name",
                queryfieldname: #N"Specification3Name"
              },
              {
                label: #N" - "
              },
              {
                source: #N"Specification3DescriptionVar",
                queryfieldname: #N"Description"
              }
              ]
},
```

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.

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

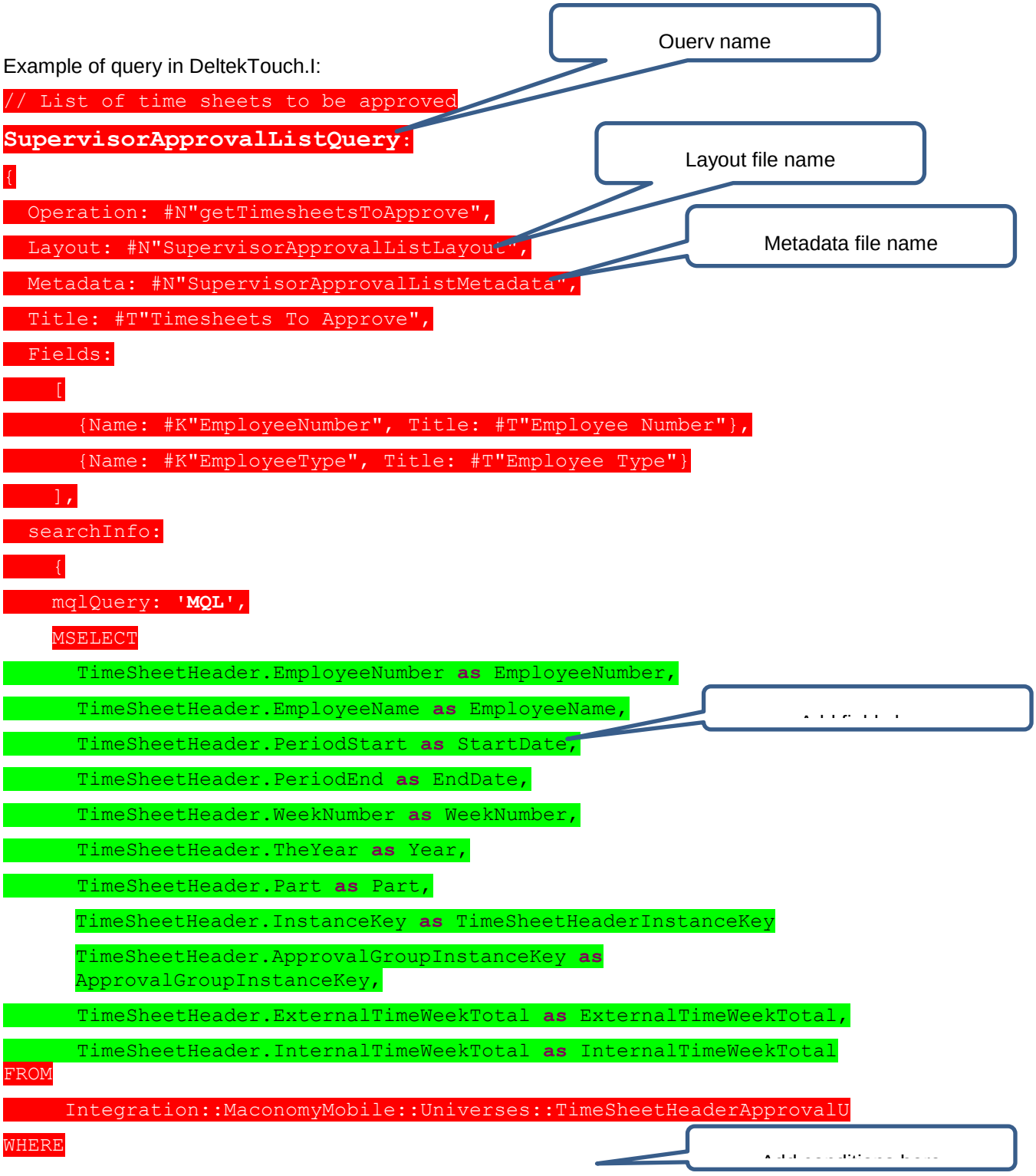
Each screen displays a list of items. These items are fetched using a query (typically to a universe; sometimes directly to the database). These queries are now available for customizations in DeltekTouch.I setup file.

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.

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

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,
      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" }
    ]
  }
},
```

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 (e.g. Timesheet Approval)
2. Identify the field you would like to add (e.g. the company = CompanyNumber field)
3. Find the custom layout to change (e.g. …\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 (e.g. SupervisorApprovalListQuery).
   The easiest is to search for the layout name (e.g. SupervisorApprovalListLayout) in the file.
6. 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
```

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:

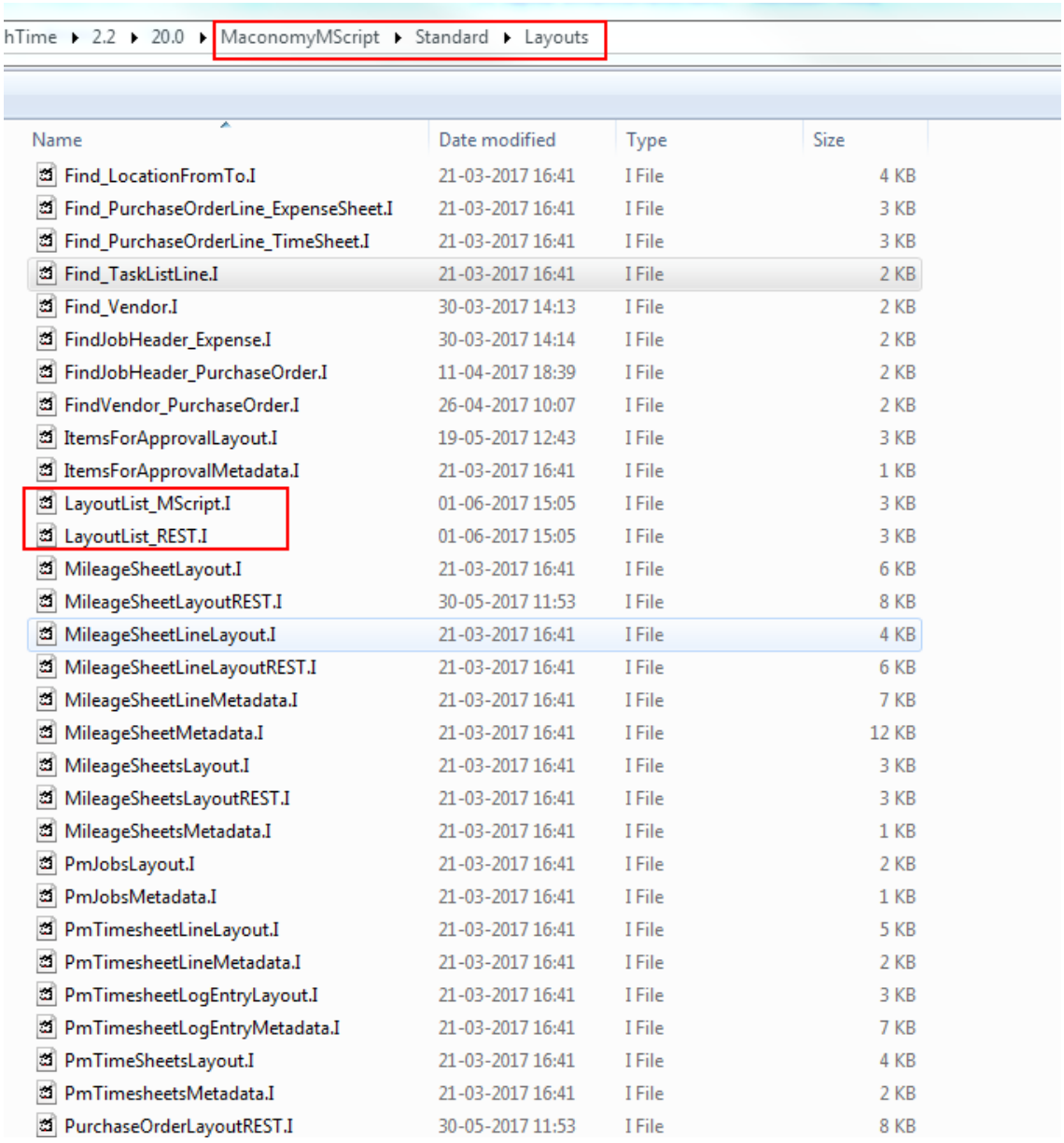
```
// Returns the screen layouts

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 (.l) files that store the same information:

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

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

The released DeltekTouch product includes standard layouts andcorresponding standard layout list files:



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.

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 Table

Time Registration

MScript	REST
TimesheetLayout.I	TimesheetLayoutREST.I TimesheetDaysLayoutREST.I
TimesheetLogEntryLayout.I	TimesheetLogEntryWeeklyLayoutREST.I

MScript	REST
	TimesheetLogEntryDailyLayoutREST.I
SummaryLayout.I	SummaryLayoutREST.I SummaryDailyLayoutREST.I
RejectedTimesheetsLayout.I	RejectedTimesheetsLayoutREST.I
-	TimeSheetWeekLayoutREST.I
-	TimeSheetCalendarLayoutREST.I
-	Find_JobHeader.I
-	Find_TaskListLine.I
-	Find_JobFavorite_Timesheet.
-	Find_JobHeader_Timesheet.I
-	Find_Activity.I

Expense and Mileage Registration

MScript	REST
ExpenseSheetsLayout.I	ExpenseSheetsLayoutREST.I
ExpenseSheetLayout.I	ExpenseSheetLayoutREST.I
ExpenseSheetLineLayout.I	ExpenseSheetLineLayoutREST.I
CreateExpenseSheetLineLayout.I	QuickCaptureExpenseSheetLineLayoutREST.I
QuickCaptureAttachReceiptLayout.I	QuickCaptureAttachReceiptLayoutREST.I
MileageSheetsLayout.I	MileageSheetsLayoutREST.I
MileageSheetLayout.I	MileageSheetLayoutREST.I
MileageSheetLineLayout.I	MileageSheetLineLayoutREST.I
-	Find_LocationFromTo.I
-	FindJobHeader_Expense.I
-	DocumentArchiveListREST.I

Approvals

MScript	REST
PmTimesheetLogEntryLayout.I	-
PmJobsLayout.I	-
PmTimeSheetsLayout.I	-
PmTimesheetLineLayout.I	-
SupervisorApprovalListLayout.I	-
SupervisorTimesheetLogEntryLayout.I	-
SupervisorTSJobLayout.I	-
SupervisorTSDayLayout.I	-
SupervisorEntriesByJobLayout.I	-
SupervisorEntriesByDayLayout.I	-
ApprovalExpenseSheetListLayout.I	-
ApprovalMileageSheetLayout.I	-
ApprovalExpenseSheetLineListLayout.I	-
ApprovalExpenseSheetLayout.I	-
ApprovalExpenseSheetLineJobListLayout.I	-
ApprovalExpenseSheetLineLayout.I	-
ApprovalExpenseSheetLineEmployeeListLayout.I	-
ApprovalMileageSheetLineLayout.I	-
ApprovalPurchaseOrderLayout.I	-
ApprovalPurchaseOrderJobListLayout.I	-
ApprovalPurchaseOrderListLayout.I	-
ApprovalPurchaseOrderLineLayout.I	-
ApprovalPurchaseOrderPMLayout.I	-
ApprovalPurchaseOrderLinePMLayout.I	-
ApprovalVendorInvoiceListLayout.I	-

MScript	REST
ApprovalVendorInvoiceLayout.I	-
ApprovalVendorInvoiceAllocationLineLayout.I	-
ApprovalVendorInvoiceAllocationLinePMLayout.I	-
ApprovalVendorInvoiceAllocationLineJobListLayout.I	-
ApprovalVendorInvoiceListPMLayout.I	-
ApprovalVendorInvoicePMLayout.I	-
ApprovalInvoiceDraftListLayout.I	-
ApprovalInvoiceDraftLayout.I	-
ApprovalInvoiceDraftLineLayout.I	-
ApprovalInvoiceDraftLinesLayout.I	-
ApprovalExpenseSheetLinePMLayout.I	-
ApprovalMileageSheetLinePMLayout.I	-
-	AllowanceApprovalFormLayoutREST.I
-	AllowanceApprovalListLayoutREST.I
-	AllowanceApproveInputScreenREST.I

Search Layouts

MScript	REST
-	Find_AbsenceType.I
-	Find_Account.I
-	Find_Activity.I
-	Find_Employee.I
-	Find_JobFavorite_Timesheet.I
-	Find_JobHeader_Timesheet.I
-	Find_LocationFromTo.I
-	Find_PurchaseOrderLine_ExpenseSheet.I

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

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

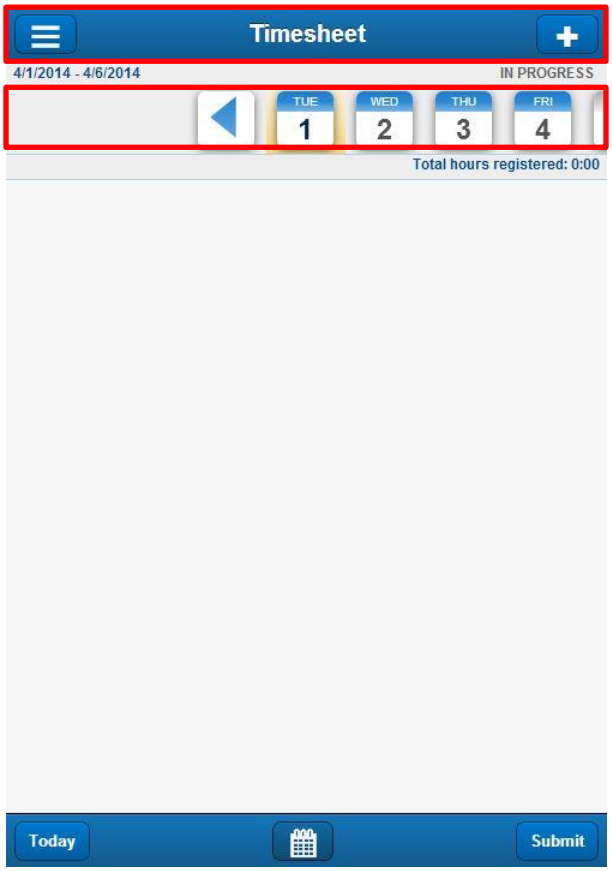
Layout Configuration

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).

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

Figure 2: 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.

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.

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 fields from a layout.

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.

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.

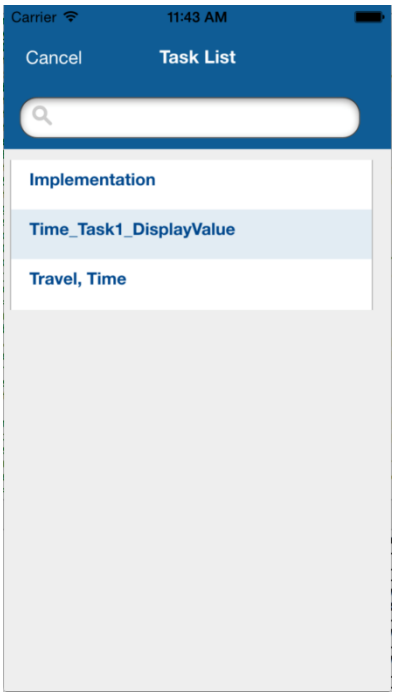
Component	Description
TextField	This component is used to display a text field on a Line screen (see Figure 1).
DocumentField	This component is used to display a field on the Line screen intended for attachments. You need to define the “DocumentName” as the source field not the “DocumentArchiveNumber” 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.
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. ▪ 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).
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

Attribute	Description

	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",`
	`label: #N"Remark",`
	`searchlayout: #N"ActivitySearch",`
	`type: #N"lookupfield",`
	`queryfieldname: #N"KeyValue",`
	`displayfields: [`
	`{`
	`source: #N"ActivityTextVar",`
	`queryfieldname: #N"DisplayValue"`
	`},`
	`{`

Attribute	Description

```
                    label: #N" – ",
                },
                {
                    source: #N"ActivityNumber",
                    queryfieldname: #N"DisplayValue"
                }
        ]
        Lookuplayoutname: "DeliverableLookupLayout"
}
```

Adding the layout to the LayoutList_MScript.IDeltekTouchWS.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");
```

Content of a Lookup Layout File:

```
{
Layoutname: #N"DeliverableLookupLayout",
Layout:   [
  {      region: #"ToolbarArea",
          protectedregion:    true
    },
  {
```

Attribute	Description

	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.
	▪ **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.l.

Attribute	Description

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”`
	`}`
	 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”`
	`}`
	 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”`
	`}`

Attribute	Description
	For more details about the value you can define, see 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”`
	`}`
	 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”`
	`}`
	 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”`
	`}`
	 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”`
	`}`

Attribute	Description
	For more details about the value you can define, see http://www.w3schools.com/cssref/pr_margin-bottom.asp.

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”`
	`}`
	 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”`
	`}`
	 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”`
	`}`

Attribute	Description
	For more details about the value you can define, see http://www.w3schools.com/cssref/css3_pr_text-overflow.asp.

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”,`
	`}`
	 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”,`
	`}`
	 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”,`
	`}`
	 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: {`

Attribute	Description
	paddingbottom: "16px", } For more details about the value you can define, see 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"
	}
	 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"
	}
	 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)"
	}
	 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"
	}

Attribute	Description
	For more details about the value you can define, see http://www.w3schools.com/cssref/css3_pr_border-radius.asp.

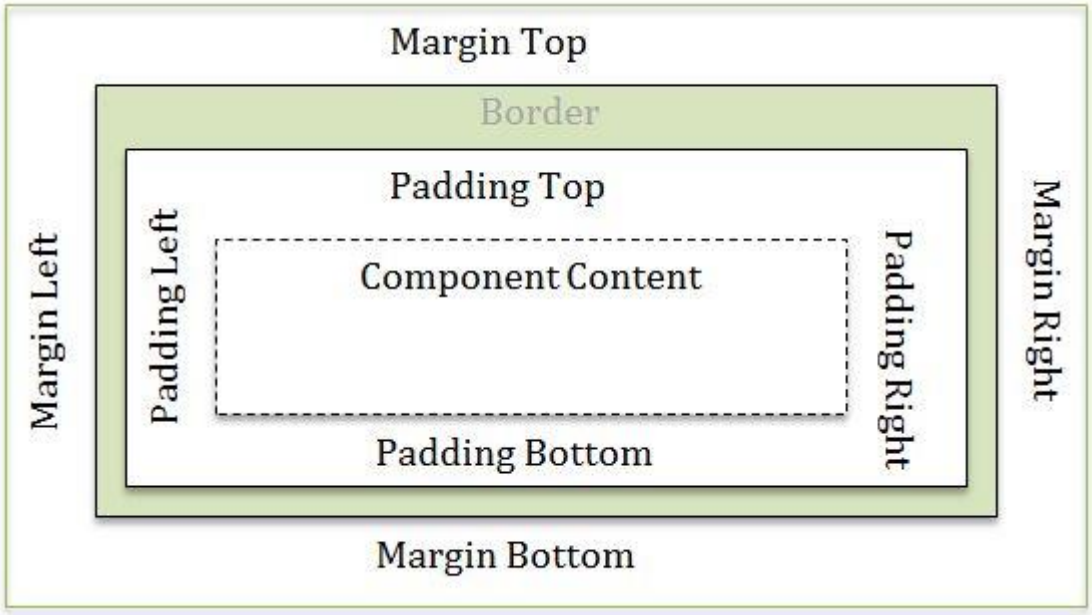
bordercolor	This is used for applying border color of an element.
	Example:
	`style: {`
	`bordercolor: #N"#ff0000 #0000ff"`
	`}`
	The border color will apply in the top and right border of an element.
	For more details about the value you can define, see http://www.w3schools.com/cssref/pr_border-color.asp.

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

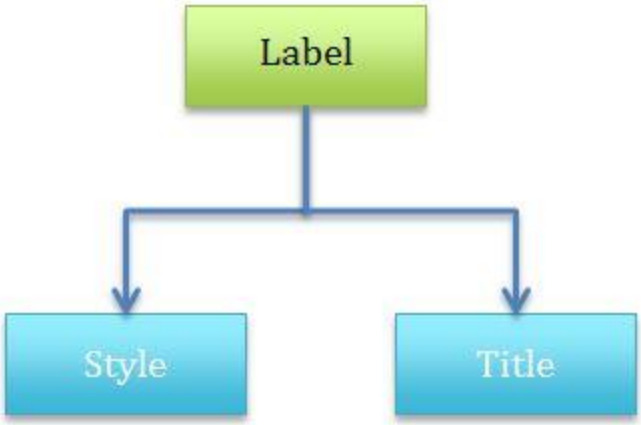
maxwidth	This is used to specify the maximum width of an element.
	Example:
	`style: {`
	`maxwidth: #N"10px"`
	`}`
	For more details about the value you can define, see http://www.w3schools.com/cssref/pr_dim_max-width.asp.

Margin and Padding Box Model



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.

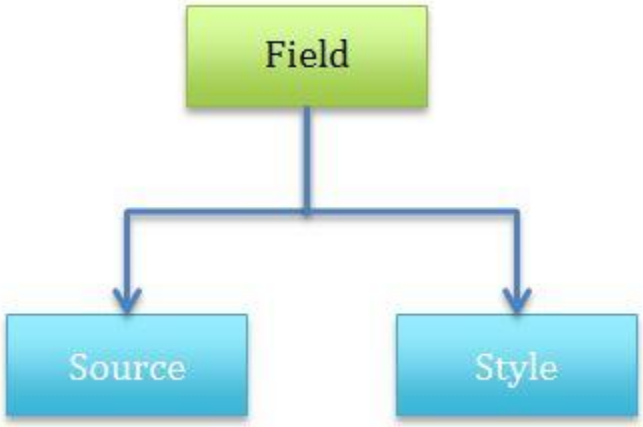
Example

-

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

- Output: Employee Number:001

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).

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

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

- 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"
  ```

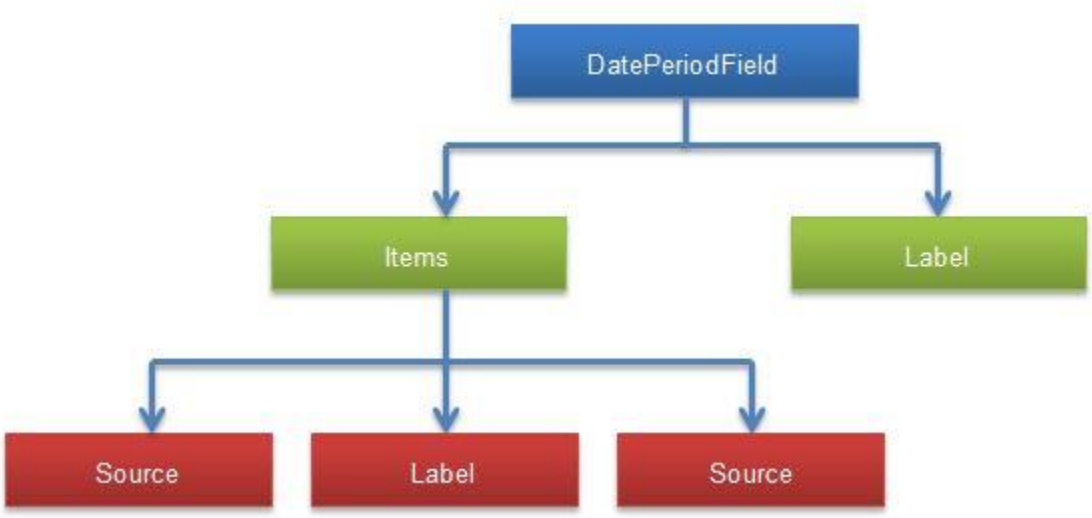
```
}
```

- 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”
}
```

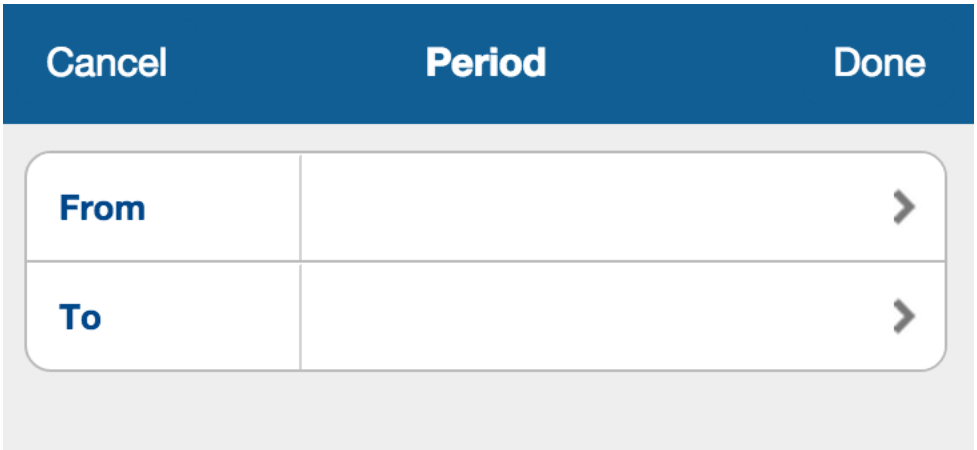
- **YYYY** represents the year (for example, 2014)
- **dd** represents the day of the month (from 00 - 31)
- **mm** represents the month of the year (from 01 – 12)

Date Period Field



Description

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



Rules

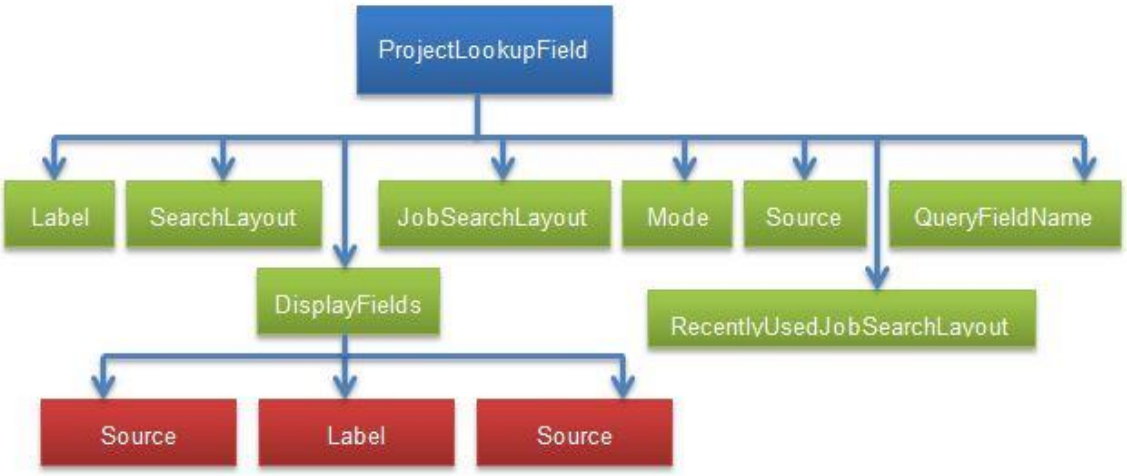
- This component does not support any styling.
- Items always contain a source, label, and another source in exact order.
- The source must be of type **Date**.
- The first label corresponds to label of the field. The second label act as a separator for the two dates.

Examples

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

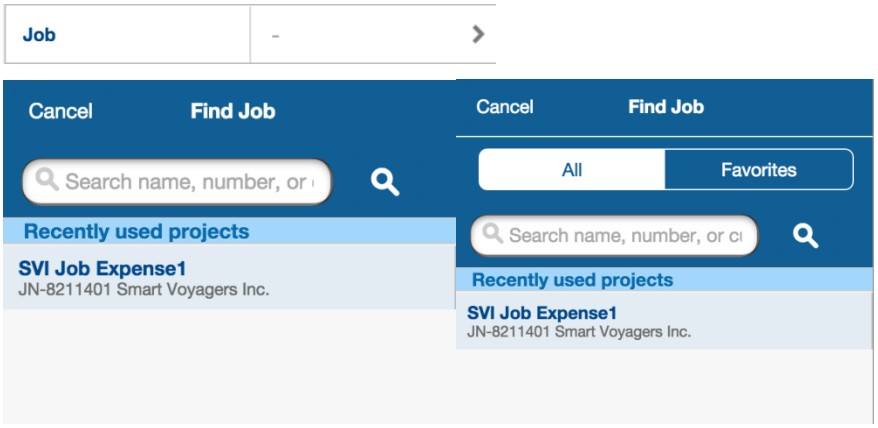
```
{
      type: “dateperiodfield”,
      label: “Period”,
      items: [
            { source: “FromDate” },
            { label: “-“ },
            { source: “ToDate” }
      ]
}
```

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.”

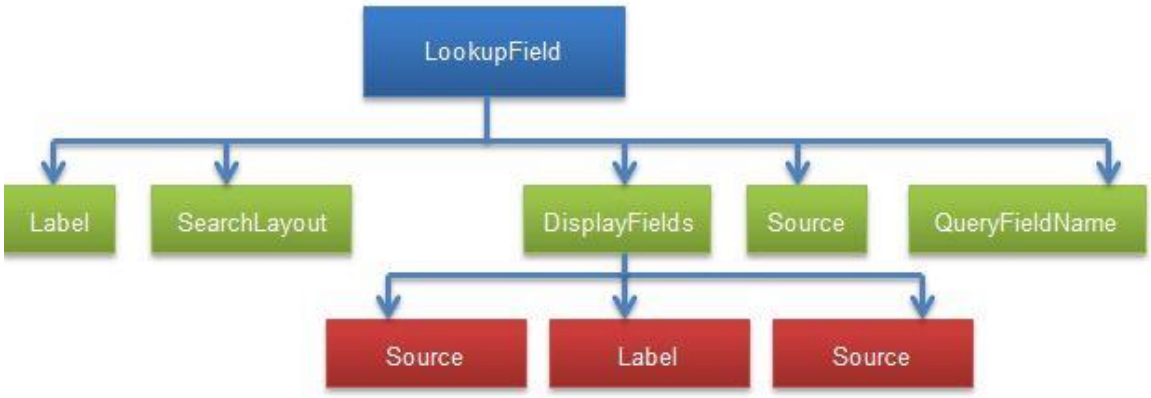
Examples

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

```
{
      source: "JobNumber",
```

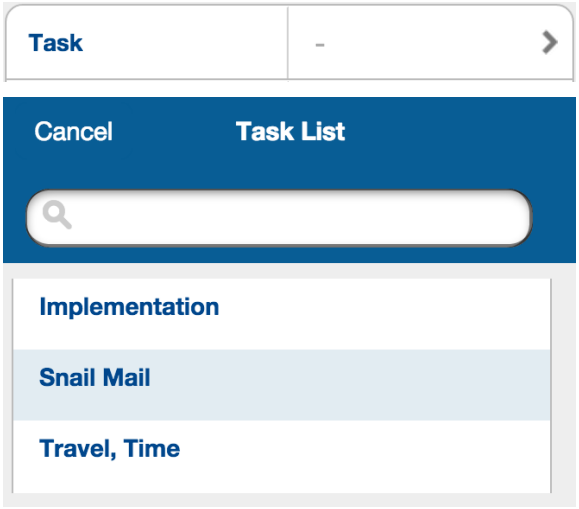
```
        label: "Job",
        searchlayout: "Expense_JobSearch",
        type: "projectlookupfield",
        mode: "projectfavorite",
        jobsearchlayout: "Expense_JobSearch",
        recentlyusedjobsearchlayout: "Expense_RecentlyUsedJobSearch"
        queryfieldname: "KeyValue",
        displayfields: [
                {
                        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.”
  -

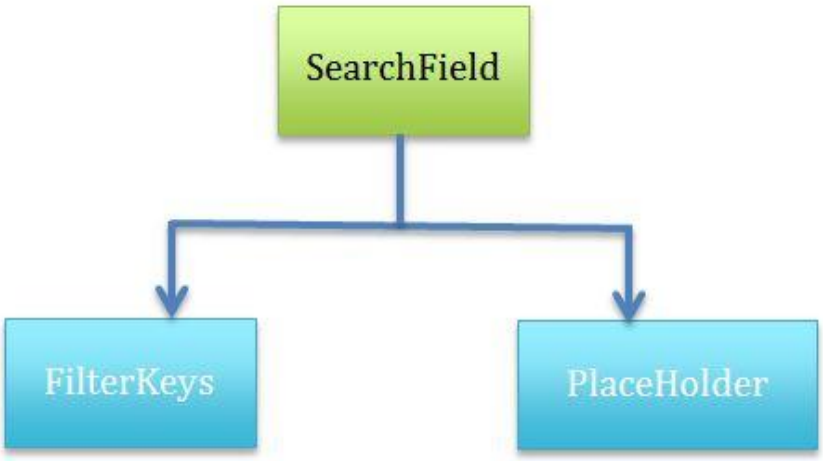
Examples

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

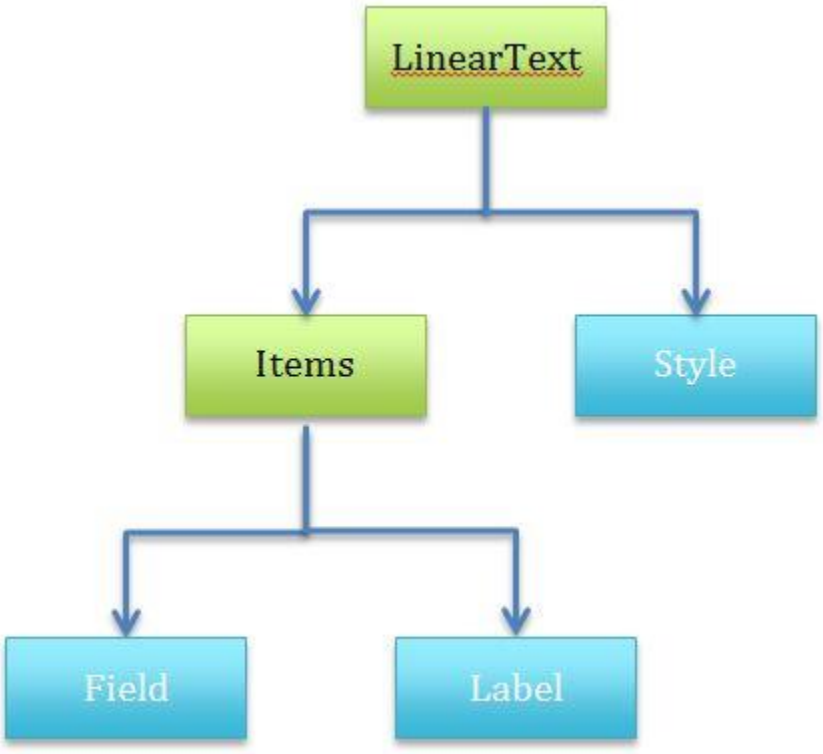
- The “filterkeys” attribute is mandatory for the searchfield component.
- Setting multiple fields in the “filterkeys” should be separated by comma character (for example, “KeyValue, DisplayValue”).
- The value that can be set in “filterkeys” is dependent on the response of the “search” operation or 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.

Examples

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

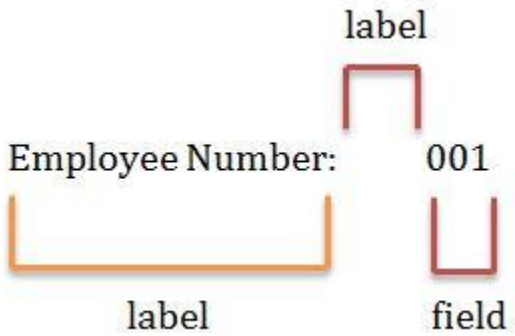
```
{
```

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

There are 10 spaces added in the following line, between "and":

```
Title: #T"          "
```

Output

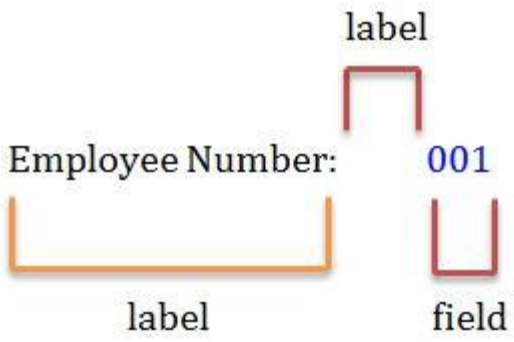


To apply "style" to the "lineartext" component (using the example above) by making the "field" item in blue, see the following example:

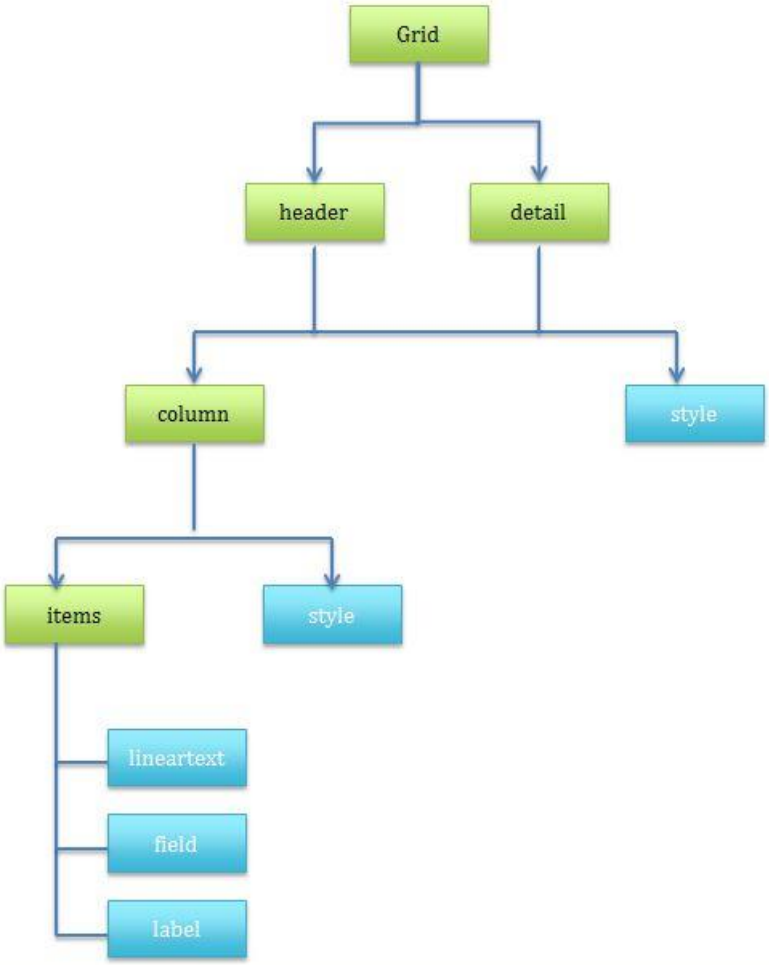
```
{
        type: "lineartext",
        style: {
                fontsize: "14px",
                fontweight: "bold"
        },
```

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

Output



Grid

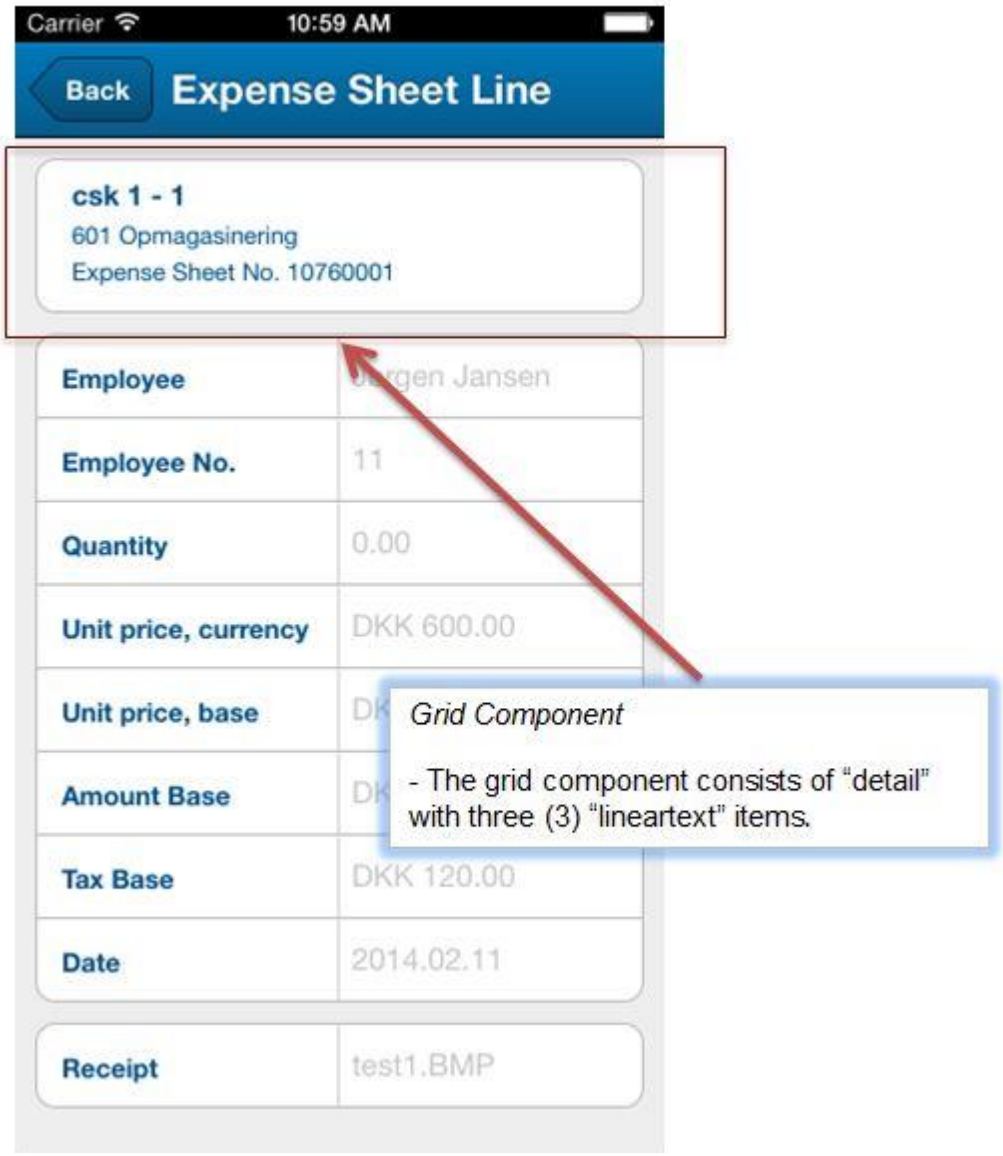


Description

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 3 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"
                                            }
```

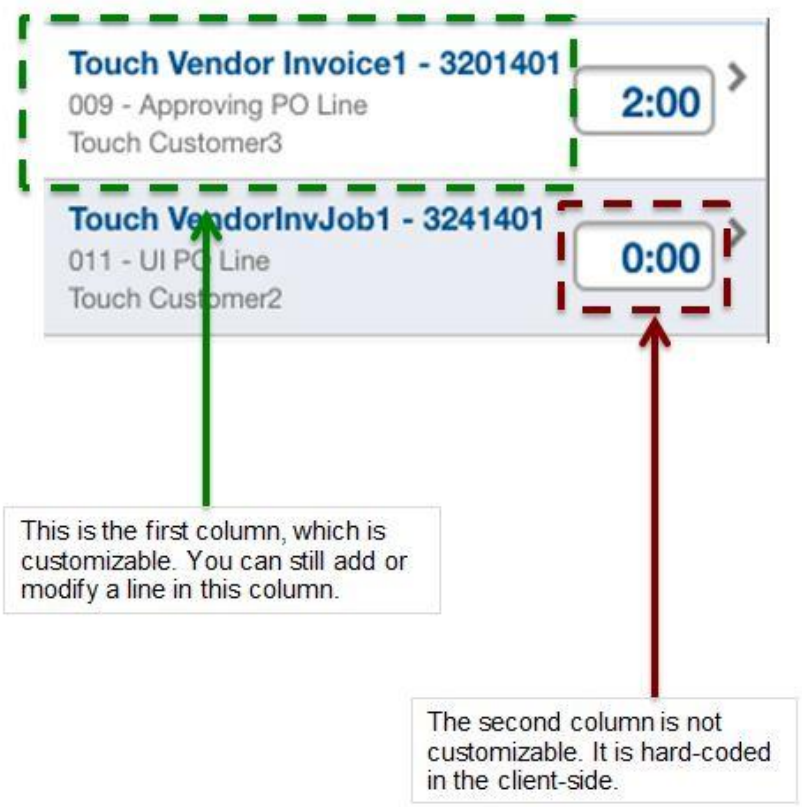
```
                                                    }
                                                }
                                            ]
                                        }
                                    ]
                                },
                                {
                                    type: “column”,
                                    region “HourField”,
                                    protectedregion: true
                                }
                            ]
}
```

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:

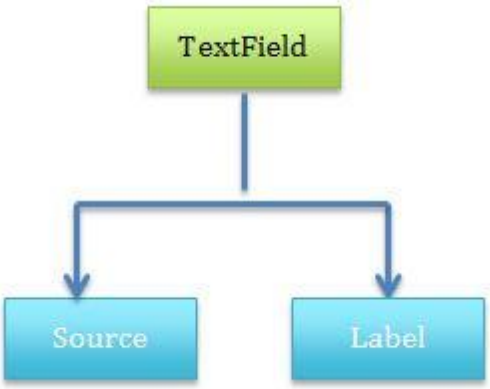
```
{
    type: “column”,
    region “HourField”,
    protectedregion: true
}
```

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

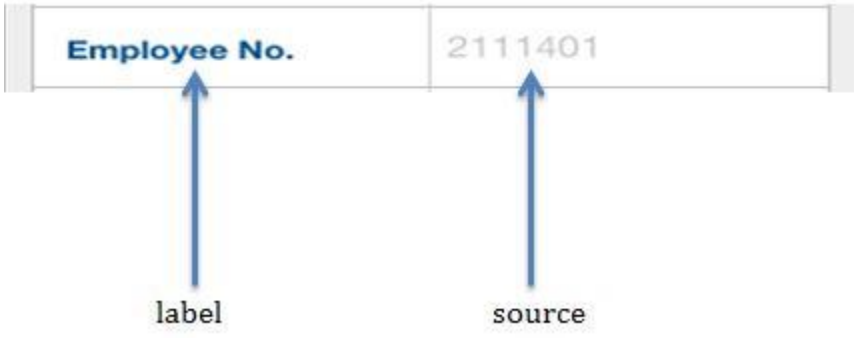
- Always specify the “source” and “label” attributes as well as the corresponding values.
- This component does not support styling. The formatting of the font used in a label is provided automatically by the client-side.

Examples

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

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

Output

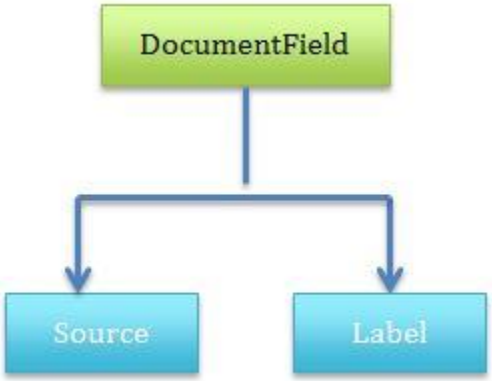


If there is a currency and an amount field that need to display in a single “textfield” component, use the “mask” attribute.

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

Output

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

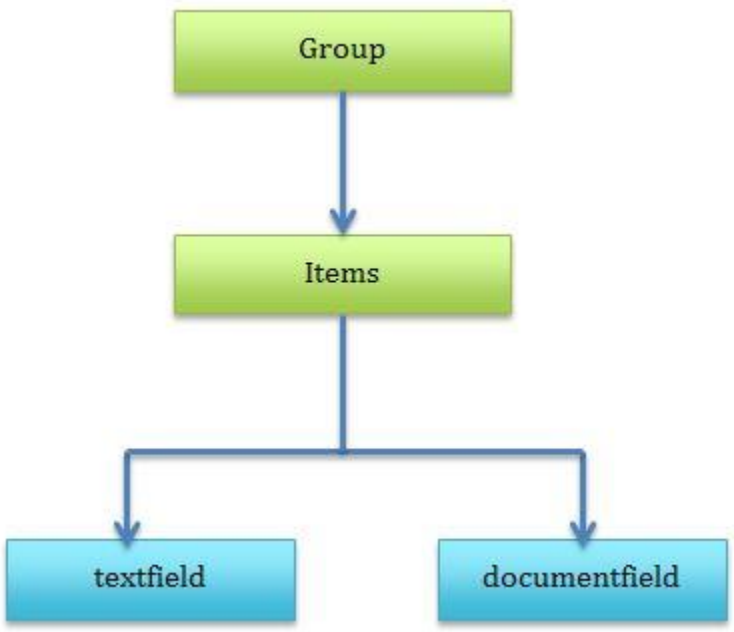
- Always specify the “source” and “label” attributes as well as the corresponding values.
- The “documentfield” component only works properly if the “source” attribute contains “DocumentName” value (see image below).

```
{
    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 4: 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.

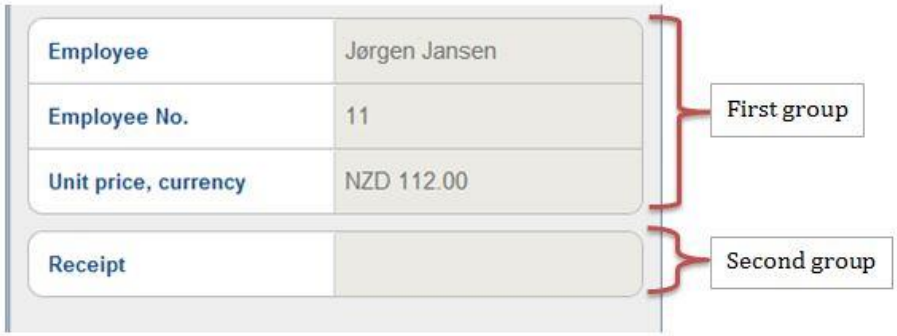
Example

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

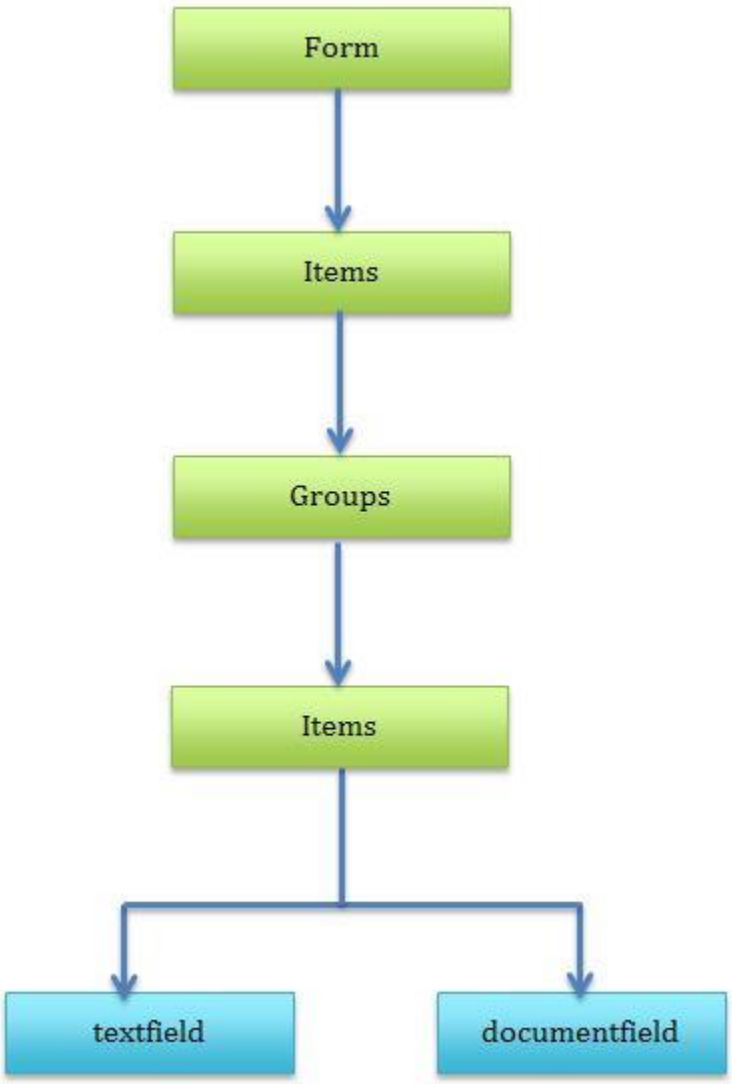
```
{
      type: “group”,
      items: [{
                  type: “textfield”,
                  source: “EmployeeNameVar”,
                  label:  “Employee”
            },
            {
```

```
                    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.

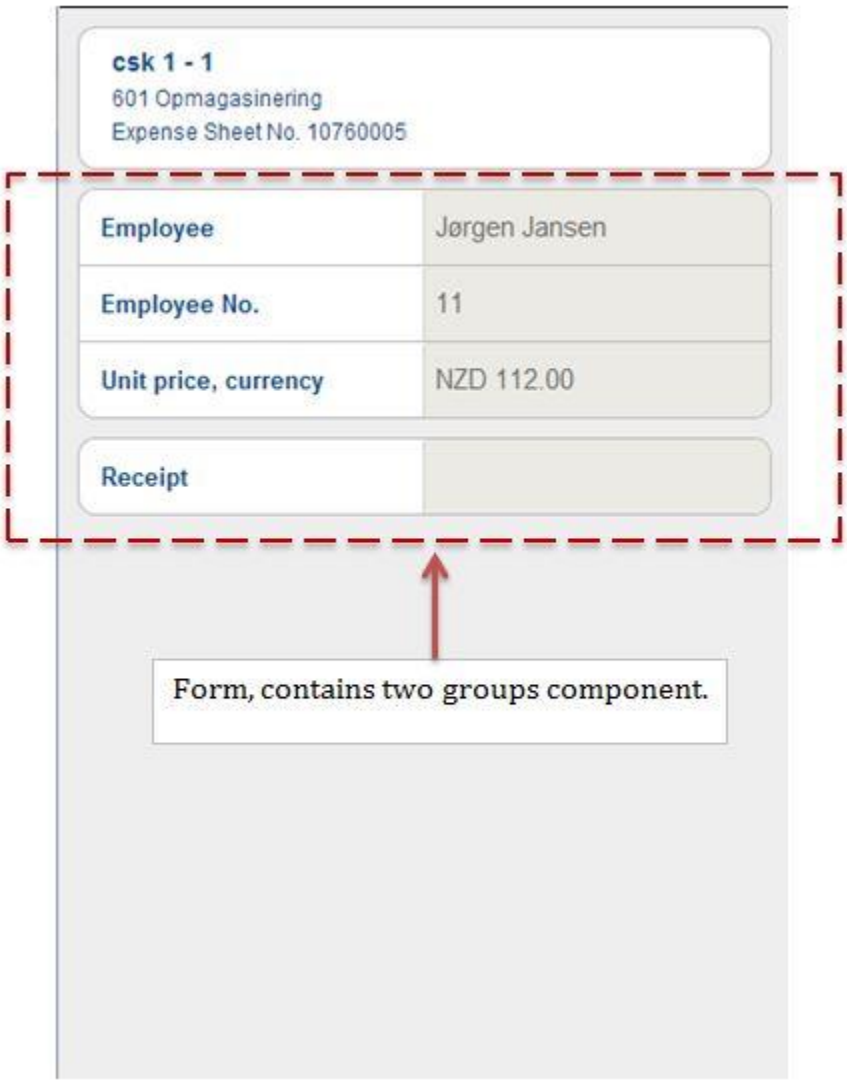
Example

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

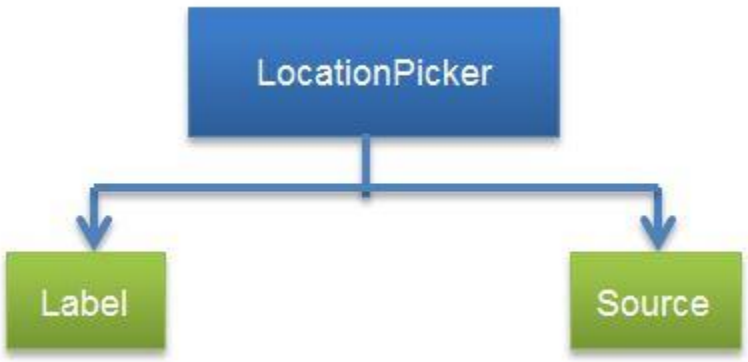
- This component should be used inside a column component.
- This component does not contain any attributes.

Examples

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

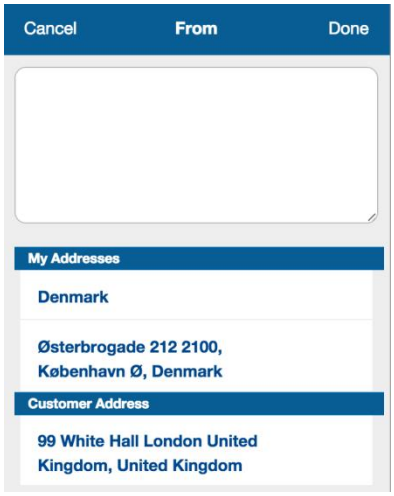
```
{
      type: “column”,
      items: [
            {
                  type: “attachmenticon”
            }
      ]
}
```

Location Picker



Description

This component is used for entering and displaying addresses.



Rules

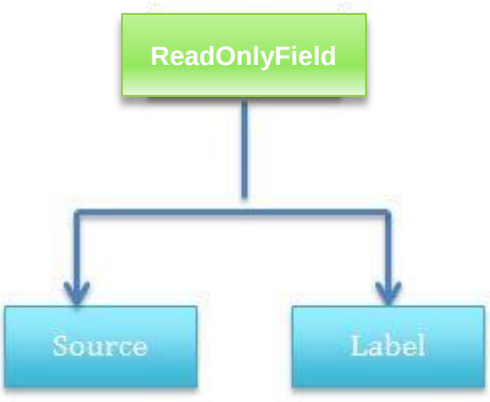
This component does not support any styling.

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:

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.

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**).

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

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",
```

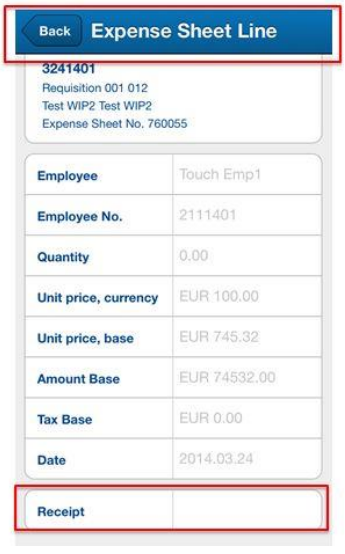
```
operation: #K"getperiod",
metadata:
   [
      // Fields from DailyTimeSheetLine
      {source: #K"ActivityNumber"},
      {source: #K"TaskName"},
      {source: #K"NumberOf"},
      {source: #K"DailyDescription"},
      {source: #K"EmployeeNumber"},
      {source: #K"TheDate"},
      {source: #K"EntryText"},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

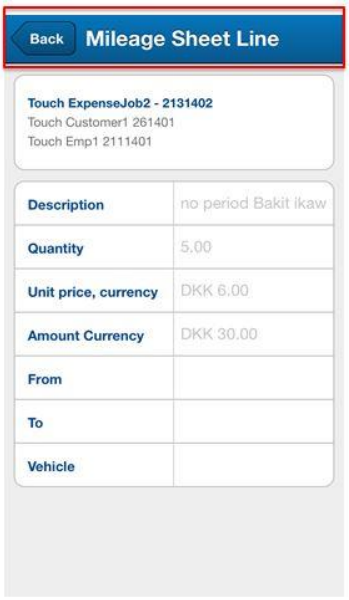
6. Change the layout by adding **LocationName** to the **RegistrationList** region in **TimesheetLayout.I**.

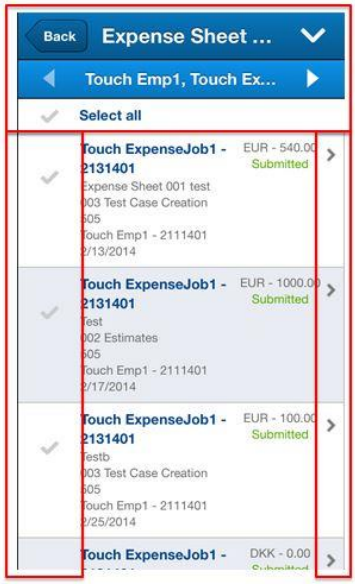
Appendix A: Screens, Layouts, and Metadata Files

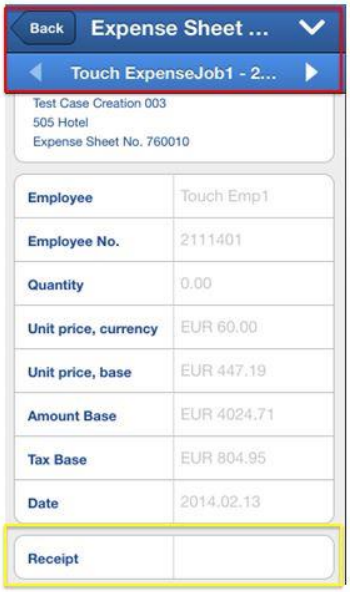
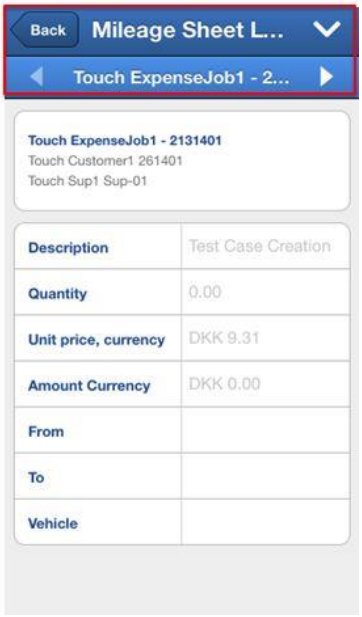
The following table displays the customizable layouts for Deltek Touch screens and their corresponding layout and metadata files:

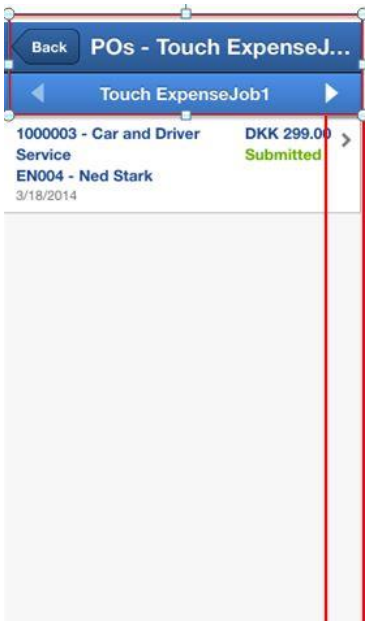
- In case your layout customization is only about formatting, the metadata file is irrelevant.
- The regions in red box are not subject for customization. Deltek recommends that you do not customize the regions in yellow box.

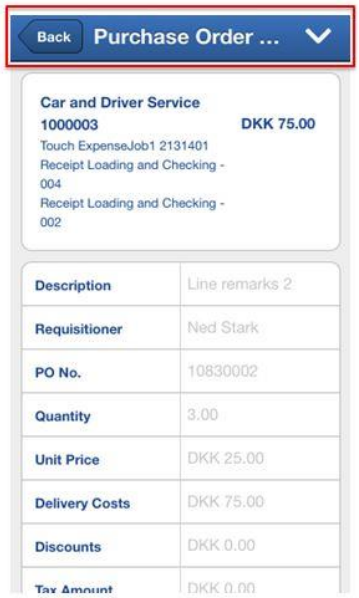
Screen	MScript	REST
	Layout: ApprovalExpenseSheetListLayout **Metadata:** ApprovalExpenseSheetListMetadata	*Not Applicable*
	**Layout:** ApprovalExpenseSheetLineLayout.l **Metadata:** ApprovalExpenseSheetLineMetadata.l	*Not Applicable*

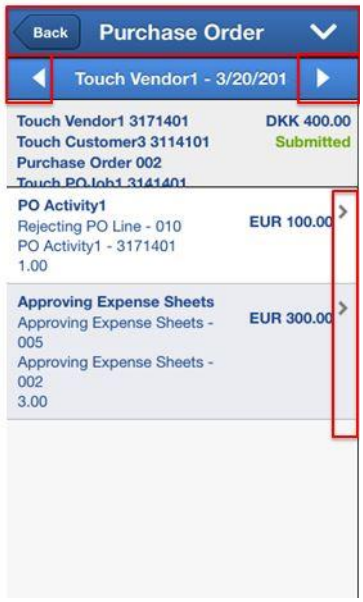
Screen	MScript	REST
	**Layout:** ApprovalMileageSheetLineLayout.l **Metadata:** ApprovalMileageSheetLineMetadata.l	*Not Applicable*
	Layout: ApprovalExpenseJobListLayout.l **Metadata:** ApprovalExpenseJobListMetadata.l	*Not Applicable*

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

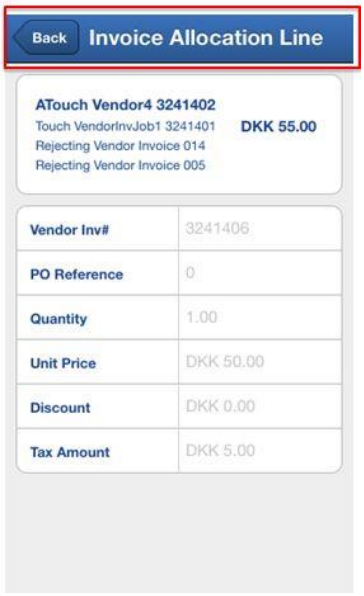
Screen	MScript	REST
	**Layout:** ApprovalExpenseSheetLineLayout.l **Metadata:** ApprovalExpenseSheetLineMetadata.l	*Not Applicable*
	**Layout:** ApprovalMileageSheetLineLayout.l **Metadata:** ApprovalMileageSheetLineMetadata.l	*Not Applicable*

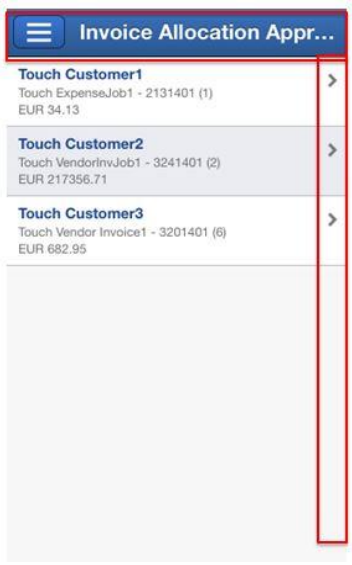
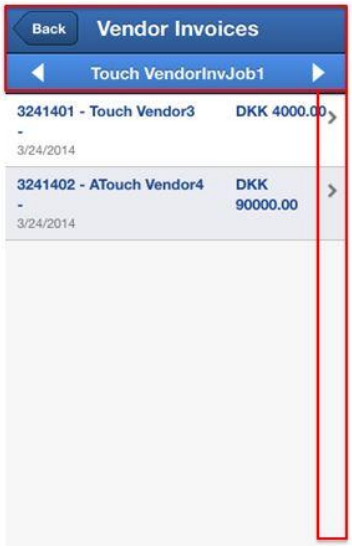
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderJobListLayout.l **Metadata:** ApprovalPurchaseOrderJobListMetadata.l	*Not Applicable*
	**Layout:** ApprovalPurchaseOrderListLayout.l **Metadata:** ApprovalPurchaseOrderListMetadata.l	*Not Applicable*

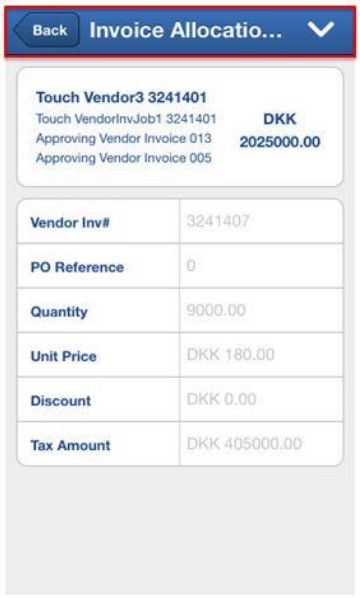
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderPMLayout.l **Metadata:** ApprovalPurchaseOrderPMMetadata.l	*Not Applicable*
	**Layout:** ApprovalPurchaseOrderLinePMLayout.l **Metadata:** ApprovalPurchaseOrderLinePMMetadata.l	*Not Applicable*

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

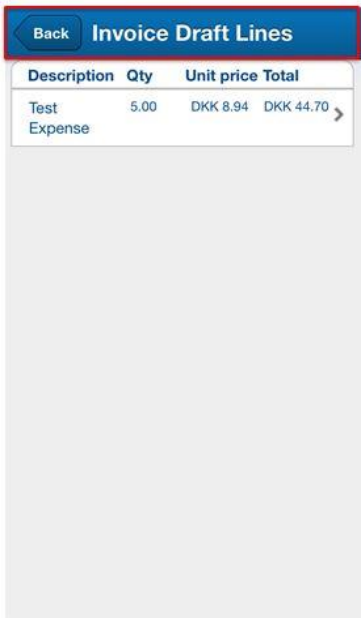
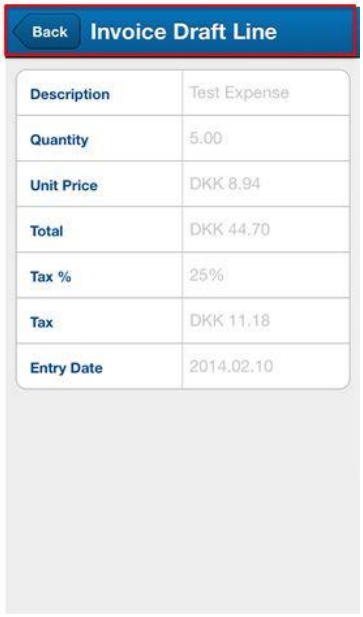
Screen	MScript	REST
	Layout: ApprovalPurchaseOrderLineLayout.l **Metadata:** ApprovalPurchaseOrderLineMetadata.l	*Not Applicable*
	Layout: ApprovalVendorInvoiceListLayout.l **Metadata:** ApprovalVendorInvoiceListMetadata.l	*Not Applicable*

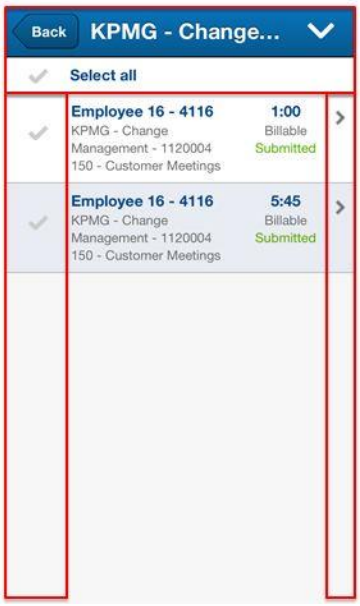
Screen	MScript	REST
	Layout: ApprovalVendorInvoiceLayout.l **Metadata:** ApprovalVendorInvoiceMetadata.l	*Not Applicable*
	**Layout:** ApprovalVendorInvoiceAllocationLineLayout.l **Metadata:** ApprovalVendorInvoiceAllocationLineMetadata.l	*Not Applicable*

Screen	MScript	REST
	**Layout:** ApprovalVendorInvoiceAllocationLineJobListLayout.I **Metadata:** ApprovalVendorInvoiceAllocationLineJobListMetadata.I	*Not Applicable*
	**Layout:** ApprovalVendorInvoiceListPMLayout.I **Metadata:** ApprovalVendorInvoiceListPMMetadata.I	*Not Applicable*

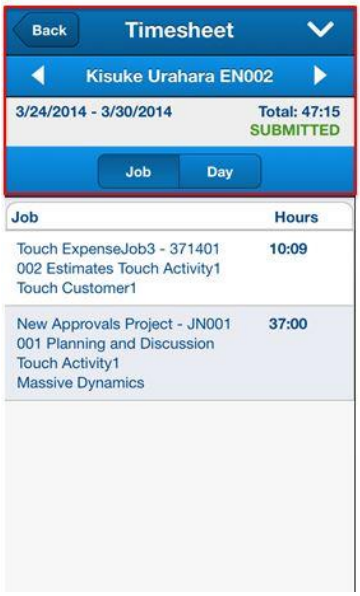
Screen	MScript	REST
	Layout: ApprovalVendorInvoicePMLayout.l **Metadata:** ApprovalVendorInvoicePMMetadata.l	*Not Applicable*
	**Layout:** ApprovalVendorInvoiceAllocationLinePMLayout.l **Metadata:** ApprovalVendorInvoiceAllocationLinePMMetadata.l	*Not Applicable*

Screen	MScript	REST
	Layout: ApprovalInvoiceDraftListLayout.l **Metadata:** ApprovalInvoiceDraftListMetadata.l	*Not Applicable*
	Layout: ApprovalInvoiceDraftLayout.l **Metadata:** ApprovalInvoiceDraftMetadata.l	*Not Applicable*

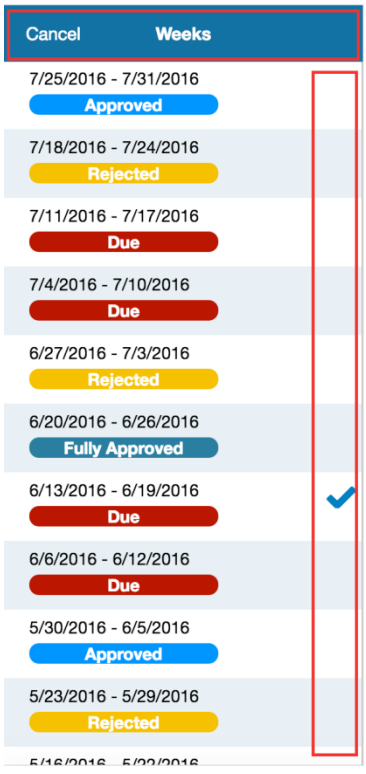
Screen	MScript	REST
	**Layout:** ApprovalInvoiceDraftLinesLayout.l **Metadata:** ApprovalInvoiceDraftLinesMetadata.l	*Not Applicable*
	**Layout:** ApprovalInvoiceDraftLineLayout.l **Metadata:** ApprovalInvoiceDraftLineMetadata.l	*Not Applicable*

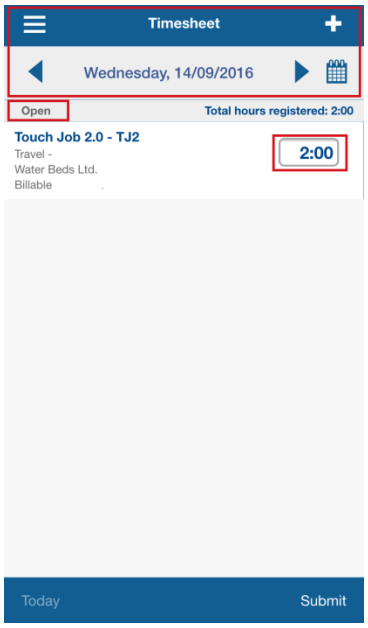
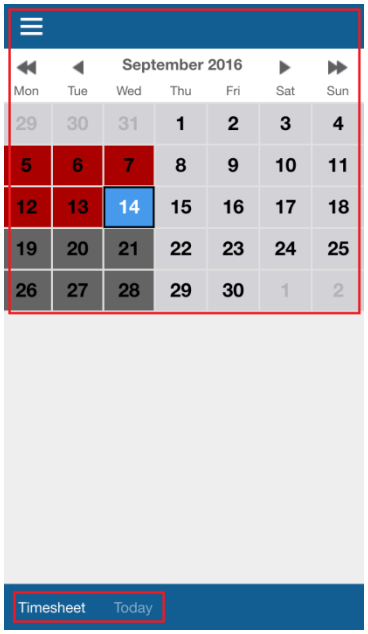
Screen	MScript	REST
	Layout: PmJobsLayout.I **Metadata:** PmJobsMetadata.I	*Not Applicable*
	**Layout:** PmTimeSheetsLayout.I **Metadata:** PmTimeSheetsMetadata.I	*Not Applicable*

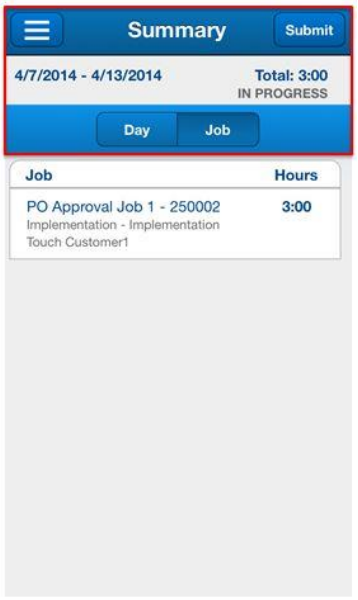
Screen	MScript	REST
	Layout: PmTimesheetLineLayout.l **Metadata:** PmTimesheetLineMetadata.l	*Not Applicable*
	Layout: SupervisorApprovalListLayout.l **Metadata:** SupervisorApprovalListMetadata.l	*Not Applicable*

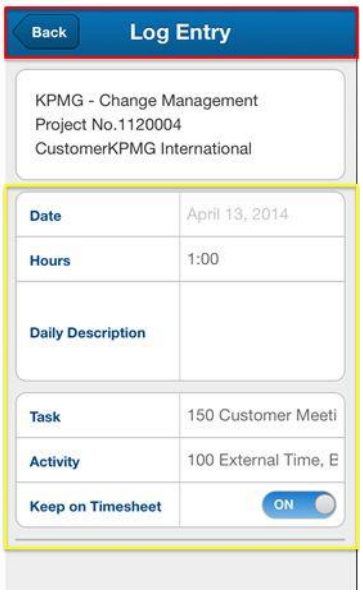
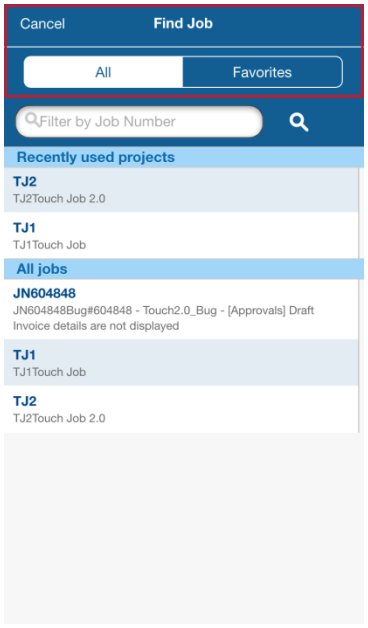
Screen	MScript	REST
	**Layout:** SupervisorTSJobLayout.I **Metadata:** SupervisorTSJobMetadata.I	*Not Applicable*
	Layout: SupervisorTSDayLayout.I **Metadata:** SupervisorTSDayMetadata.I	*Not Applicable*

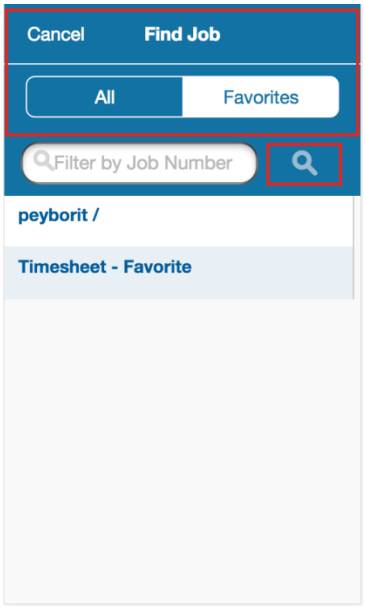
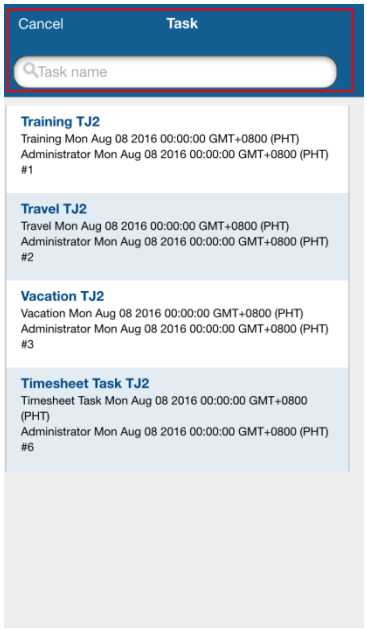
Screen	MScript	REST
	Layout: SupervisorEntriesByJobLayout.l **Metadata:** SupervisorEntriesByJobMetadata.l	*Not Applicable*
	Layout: SupervisorEntriesByDayLayout.l **Metadata:** SupervisorEntriesByDayMetadata.l	*Not Applicable*

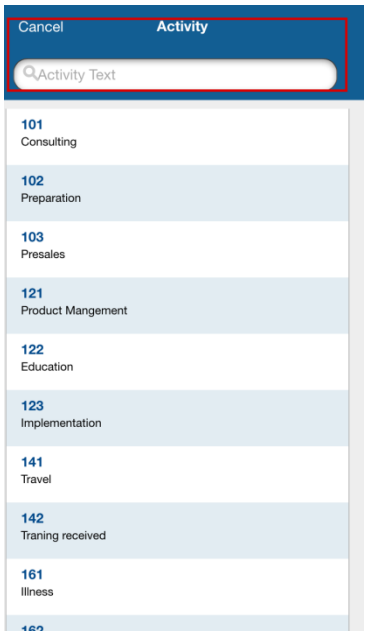
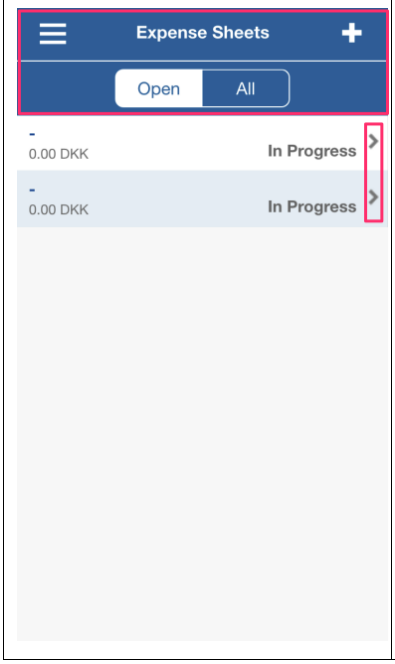
Screen	MScript	REST
	Layout: TimesheetLayout.I **Metadata:** TimesheetMetadata.I *Weekly Mode	**Layout:** TimesheetLayoutREST.I **Metadata:** None *Weekly Mode
	*Not Applicable*	**Layout:** TimeSheetWeekLayoutREST.I **Metadata:** None

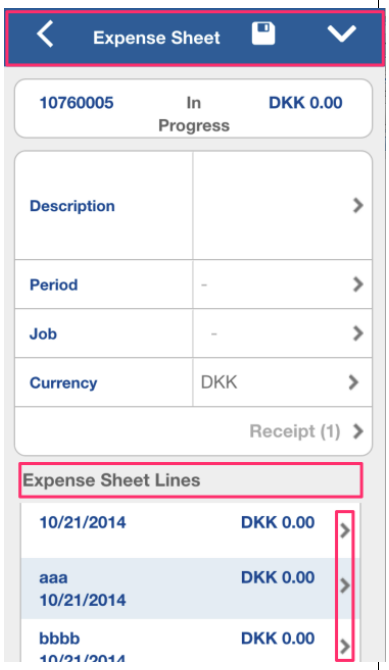
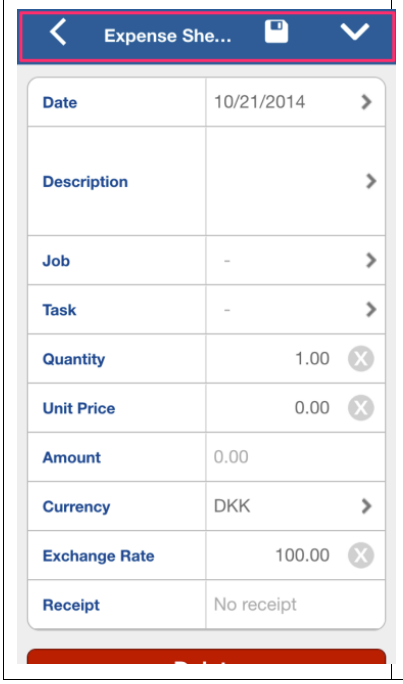
Screen	MScript	REST
	**Layout:** TimesheetLayout.I **Metadata:** TimesheetMetadata.I * Daily Mode	**Layout:** TimesheetDaysLayoutREST.I **Metadata:** None * Daily Mode
	*Not Applicable*	**Layout:** TimeSheetCalendarLayoutREST.I *Only the colors for each status are customizable.* **Metadata:** None

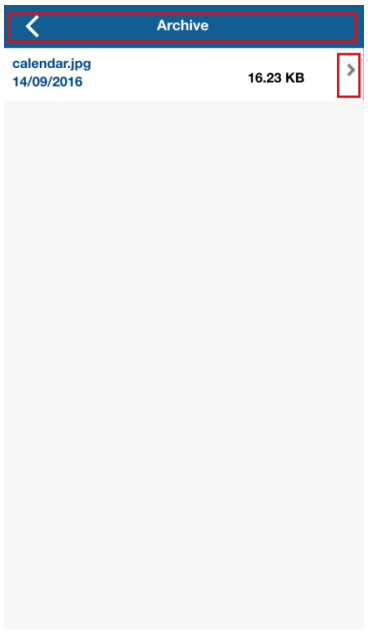
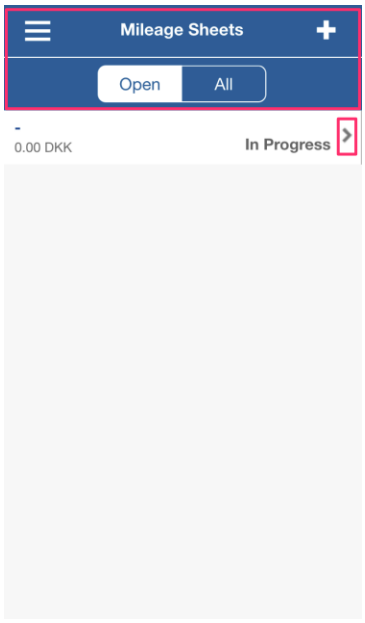
Screen	MScript	REST
  	**Layout:** SummaryLayout.I **Metadata:** SummaryMetadata.I	**Layout:** Weekly Mode - SummaryLayoutREST.I Daily Mode - SummaryDailyLayoutREST.I **Metadata:** None

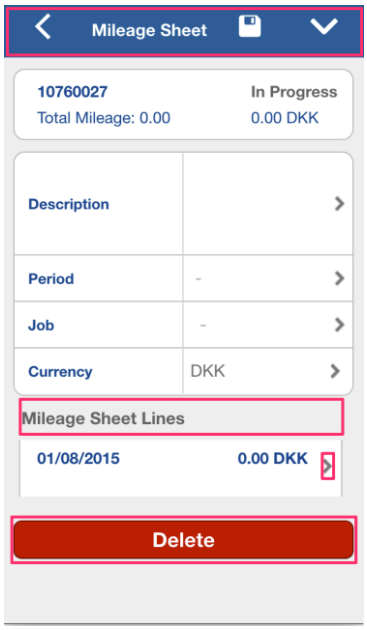
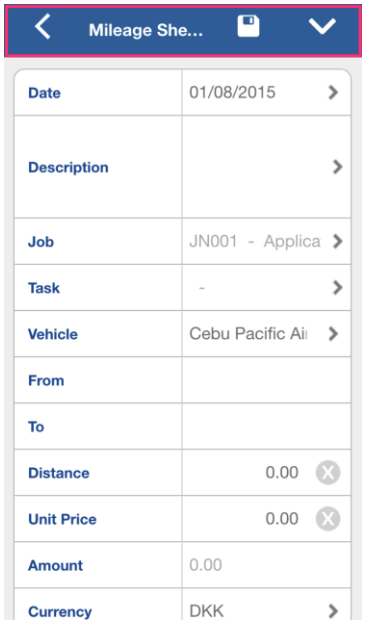
Screen	MScript	REST
	**Layout:** TimesheetLogentryLayout.I, DeltekTouch.I ▪ The new layout framework does not apply to the region in yellow box. It uses the old solution, which uses DeltekTouch.I. ▪ Three screens uses the same layout: Project Manager Timesheet Approval, Supervisor Timesheet Approval, and Timesheet **Metadata:** TimesheetLogEntryMetadata.I.	**Layout:** Weekly Mode - TimesheetLogEntryWeeklyLayoutREST.I Daily Mode - TimesheetLogEntryDailyLayoutREST.I, **Metadata:** None
	*Not Applicable*	**Layout:** Find_JobHeader_Timesheet.I **Metadata:** None

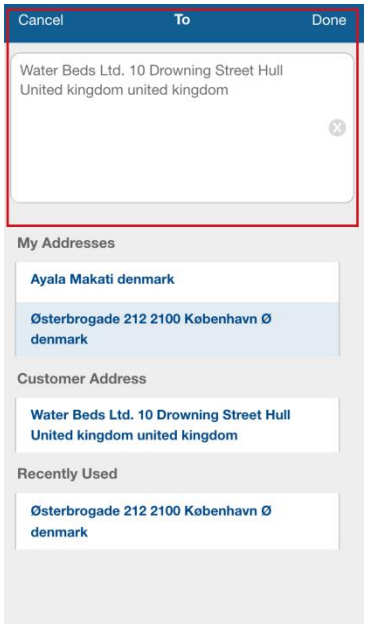
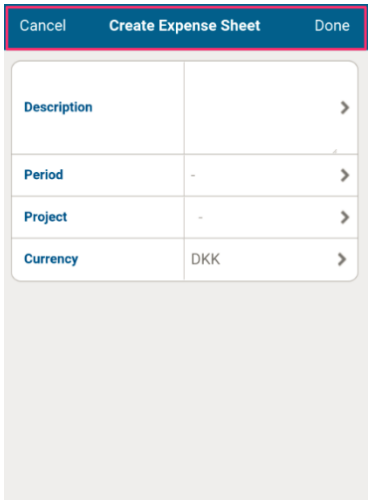
Screen	MScript	REST
	*Not Applicable*	**Layout:** Find_JobFavorite_Timesheet.l **Metadata:** None
	*Not Applicable*	**Layout:** Find_TaskListLine.l **Metadata:** None

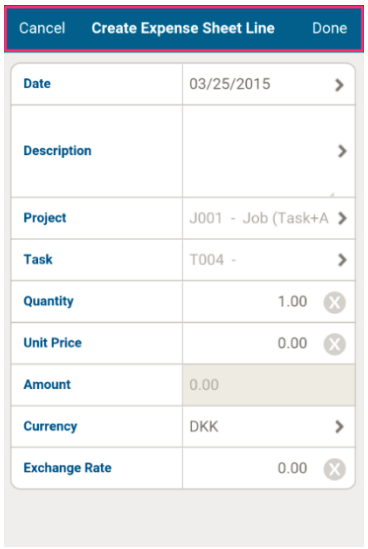
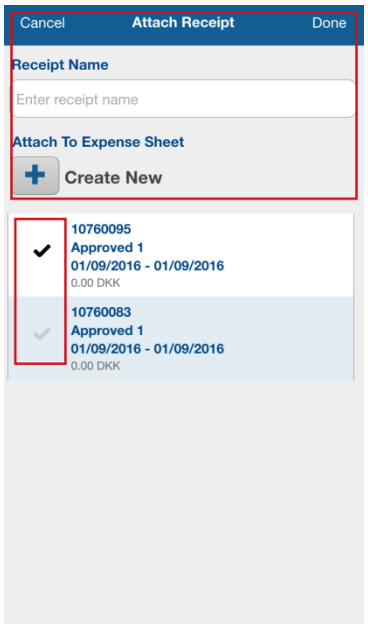
Screen	MScript	REST
	*Not Applicable*	**Layout:** Find_Activity.I **Metadata:** None
	**Layout:** ExpenseSheetsLayout.I **Metadata:** ExpenseSheetsMetadata.I	**Layout:** ExpenseSheetsLayoutREST.I **Metadata:** None

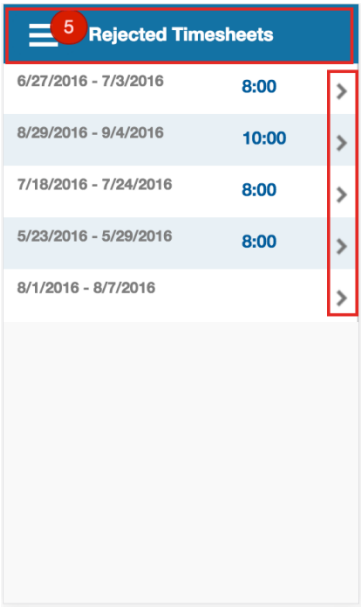
Screen	MScript	REST
	**Layout:** ExpenseSheetLayout.I **Metadata:** ExpenseSheetMetadata.I	**Layout:** ExpenseSheetLayoutREST.I **Metadata:** None
	**Layout:** ExpenseSheetLineLayout.I **Metadata:** ExpenseSheetLineMetadata.I	**Layout:** ExpenseSheetLineLayoutREST.I **Metadata:** None

Screen	MScript	REST
	*Not Applicable*	**Layout:** DocumentArchiveListREST.l **Metadata:** None
	**Layout:** MileageSheetsLayout.l **Metadata:** MileageSheetsMetadata.l	**Layout:** MileageSheetsLayoutREST.l **Metadata:** None

Screen	MScript	REST
	**Layout:** MileageSheetLayout.I **Metadata:** MileageSheetMetadata.I	**Layout:** MileageSheetLayoutREST.I **Metadata:** None
	**Layout:** MileageSheetLineLayout.I **Metadata:** MileageSheetLineMetadata.I	**Layout:** MileageSheetLineLayoutREST.I **Metadata:** None

Screen	MScript	REST
	*Not Applicable*	**Layout:** Find_LocationFromTo.l **Metadata:** None
	**Layout:** CreateExpenseSheetLayout.l **Metadata:** CreateExpenseSheetMetadata.l	*Not Applicable*

Screen	MScript	REST
	**Layout:** CreateExpenseSheetLineLayout.I **Metadata:** CreateExpenseSheetLineMetadata.I	**Layout:** QuickCaptureExpenseSheetLineLayoutREST.I **Metadata:** None
	*Not Applicable*	**Layout:** QuickCaptureAttachReceiptLayoutREST.I **Metadata:** None

Screen	MScript	REST
	**Layout:** RejectedTimesheetsLayout.l **Metadata:** RejectedTimesheetsMetadata.l	**Layout:** RejectedTimesheetsLayoutREST.l **Metadata:** None

Appendix B: For Additional Information

Customer Care Connect Site

The Deltek Customer Care Connect site 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 Customer Care Connect site 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 Connect Customer Forums
- 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
- Use Quick Chat to submit a question to a Customer Care analyst online

For more information regarding Deltek Customer Care Connect, refer to the online help available from the Web site.

Access Customer Care Connect

To access the Customer Care Connect site, complete the following steps:

1. Go to http://support.deltek.com.
2. Enter your Customer Care Connect **Username** and Password.
3. Click Log In.

If you forget your username or password, you can click the **Account Assistance** button on the login screen for help.

Available Documentation for this Release

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 Customer Care Connect site.

Document Name	Description
Deltek Touch for Maconomy Installation Guide	This document provides instructions for the installation and configuration of application.

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

Deltek is the leading global provider of enterprise software and information solutions for professional services firms, government contractors, and government agencies. For decades, we have delivered actionable insight that empowers our customers to unlock their business potential. Over 14,000 organizations and 1.8 million users in approximately 80 countries around the world rely on Deltek to research and identify opportunities, win new business, optimize resource, streamline operations, and deliver more profitable projects. Deltek – Know more. Do more.®

deltek.com