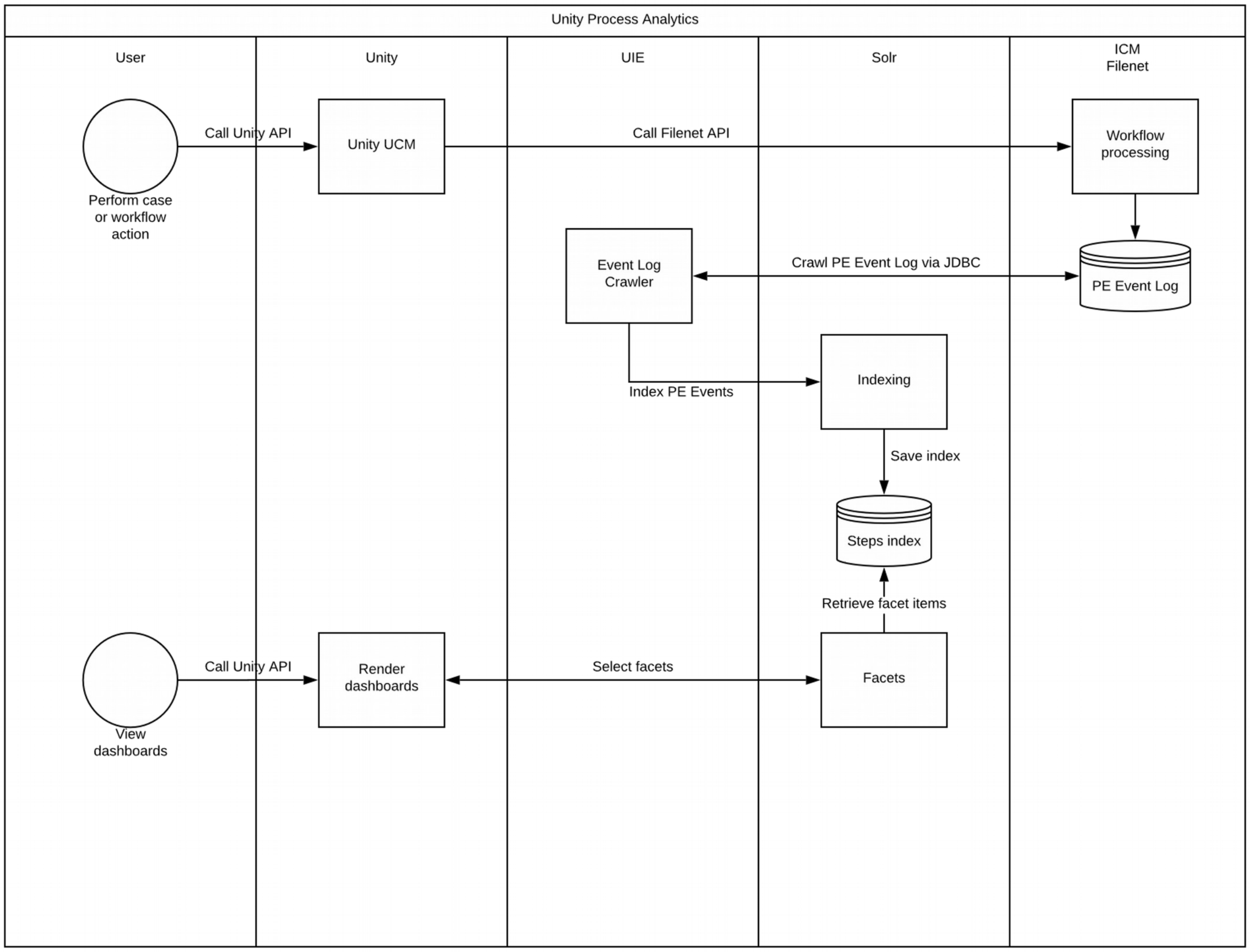
Overview

Unity Analytics module brings the ability to add leading-edge visualization and powerful analytics capabilities to the web applications based on Unity platform. It allows to summarize, aggregate, analyze and visualize various kind of data to discover insights and make more informed decisions.

The goal is not only to provide a set of pre-defined analytics screens but to provide an extensible and configurable analytics platform for rapid development of analytics interfaces to help respond to specific business questions (e.g. how many policies were created and processed last year, where are the bottlenecks in the order approval process, how claims are allocated by regions, ...)

Unity Process Analytics



Infrastructure

Unity 7.6 implementation of the Process Analytics feature relies on the following components:

- **IBM Case Manager (ICM) 5.2.1** or higher: ICM uses IBM Case Foundation (basically, it should be called FileNet Process Engine) component to automate workflows running under Case Tasks. Current implementation relies on Process Engine data model including the event logs, rosters and queues to crawl historical and real-time workflow data, as well as information about cases and tasks stored in ICM's target object store
- **Apache Solr 7.7 (Solr)** or higher: Process Analytics Unity backend uses UIE analytics capabilities which are platform-abstract by design but for now it works against Apache Solr engine.

Following top-level steps required to enable and configure Unity Process Analytics feature against some business application built on ICM and Solr:

- Design and deploy ICM solution which reflect the business application required
- Set up Unity (or Unity for Salesforce) configuration to work with the solution [NOT COVERED HERE]
- Configure FileNet Process Engine to expose the data
- Set up Apache Solr to handle the target index with analytics data model [NOT COVERED HERE]
- Configure UIE crawlers to fetch the data from ICM (FileNet): historical and real-time process data, information about cases and tasks
- Configure Process Analytics dashboards to display required analytic views to the user
- Run everything and enjoy the result.

Solution design

Unity 7.6 Process Analytics feature uses OOTB capabilities of IBM Case Manager and FileNet Process Engine (called IBM Case Foundation) underneath to extract process data. Corresponding standard solution design approach is expected to be used as well.

Public documentation describing solution design aspects are available here: IBM Knowledge Center: IBM Case Manager - Designing your case management solution and application.

In general, the core part of the Case Management solution is workflow design. To focus on that the following documentation articles would be most useful:

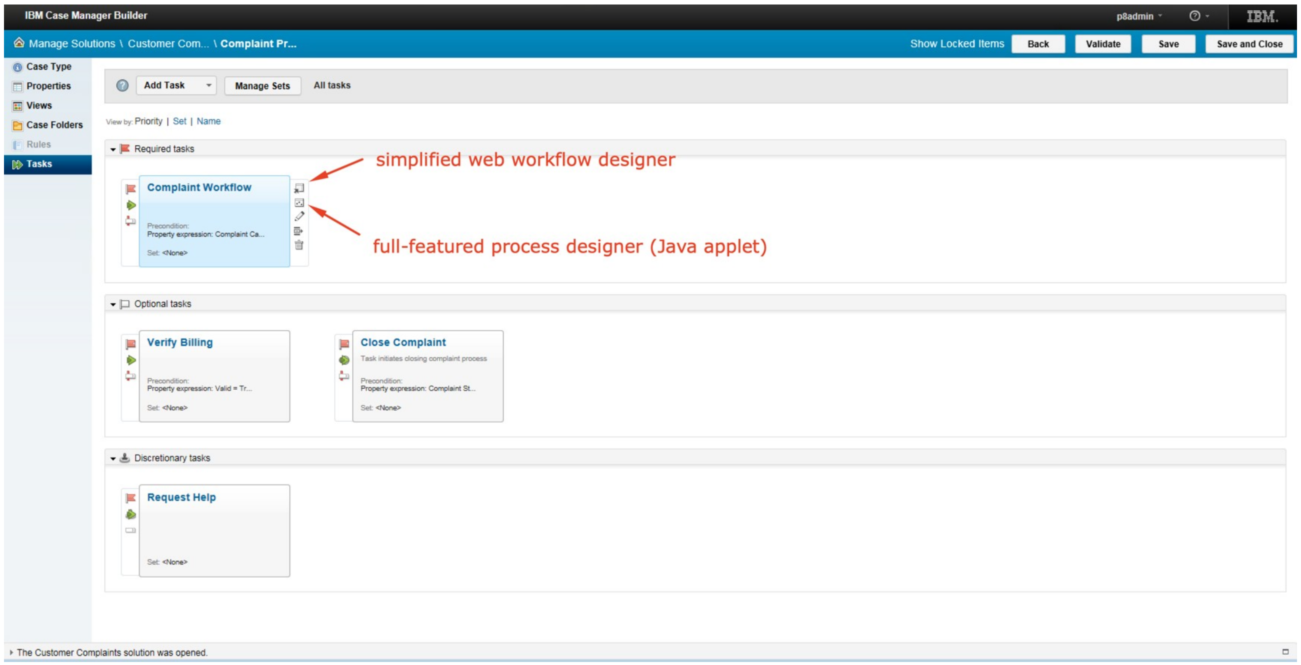
- Adding a IBM Case Foundation process as a task
- Enhancing solution designs by using Process Designer

Note, that implementing Case tasks with IBM Business Process Manager is not supported by Process Analytics due to the fact that it is an external product for the FileNet platform

Corresponding online courses and certification programs are available on IBM Skills Gateway (ex- Learning Portal) as well.

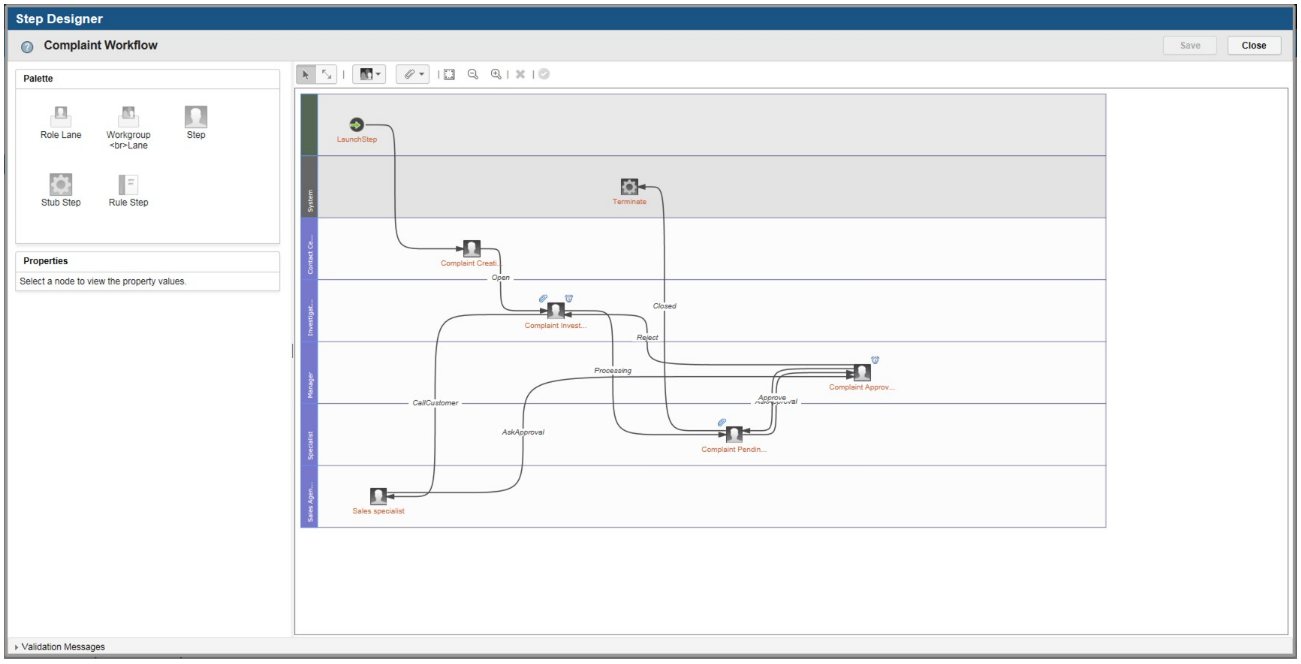
How to launch step or process designer

Here's how to design Case task's workflow within ICM Case Builder:

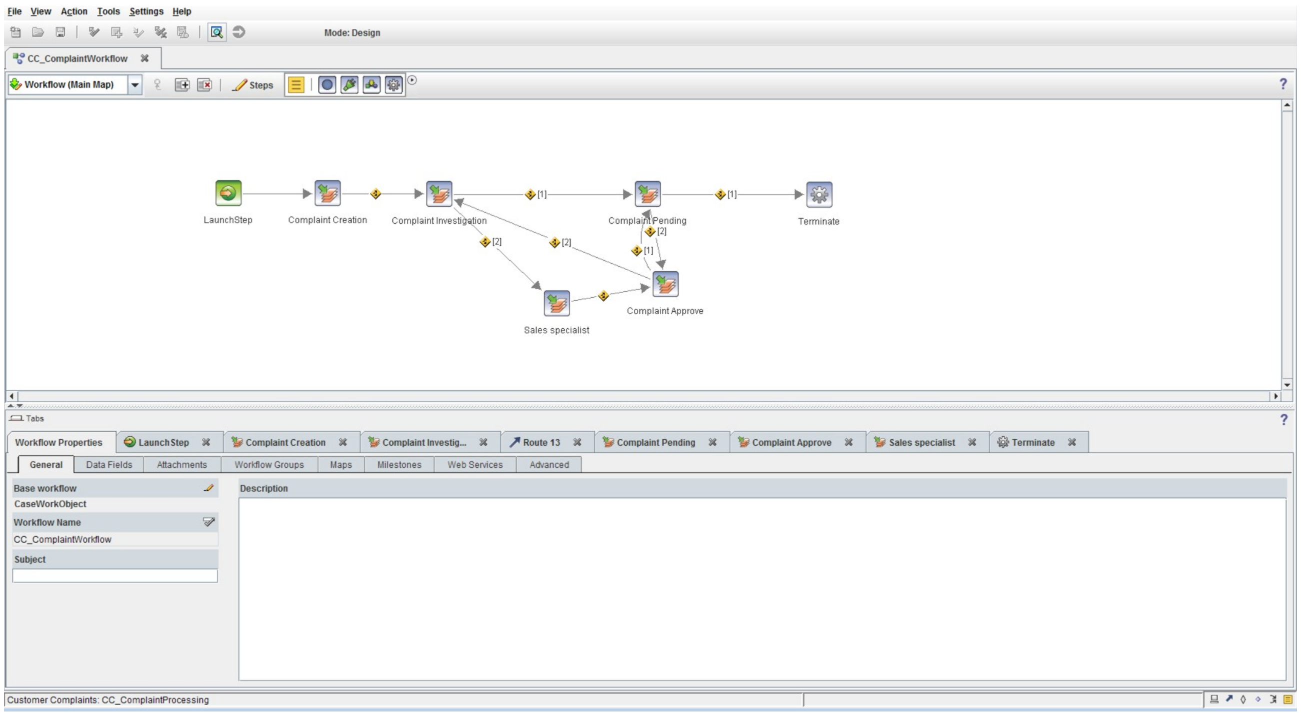


The Customer Complaints solution was opened.

The first button shown below is intended to open built-in web workflow designer (called "Step Designer"), which has only basic process development capabilities, but it doesn't require any client software other than browser to run.



Second one button launches classic Java applet called Process Designer, which is full featured FileNet tool to design processes. It allows to define dynamic workflow behavior, such as calling custom process components, using variables and expressions and so on. Since it’s a java applet, it requires an old version of browser, such as IE 11 or some old build of Firefox.



Base workflow
CaseWorkObject
Workflow Name
CC_ComplaintWorkflow
Subject
Description

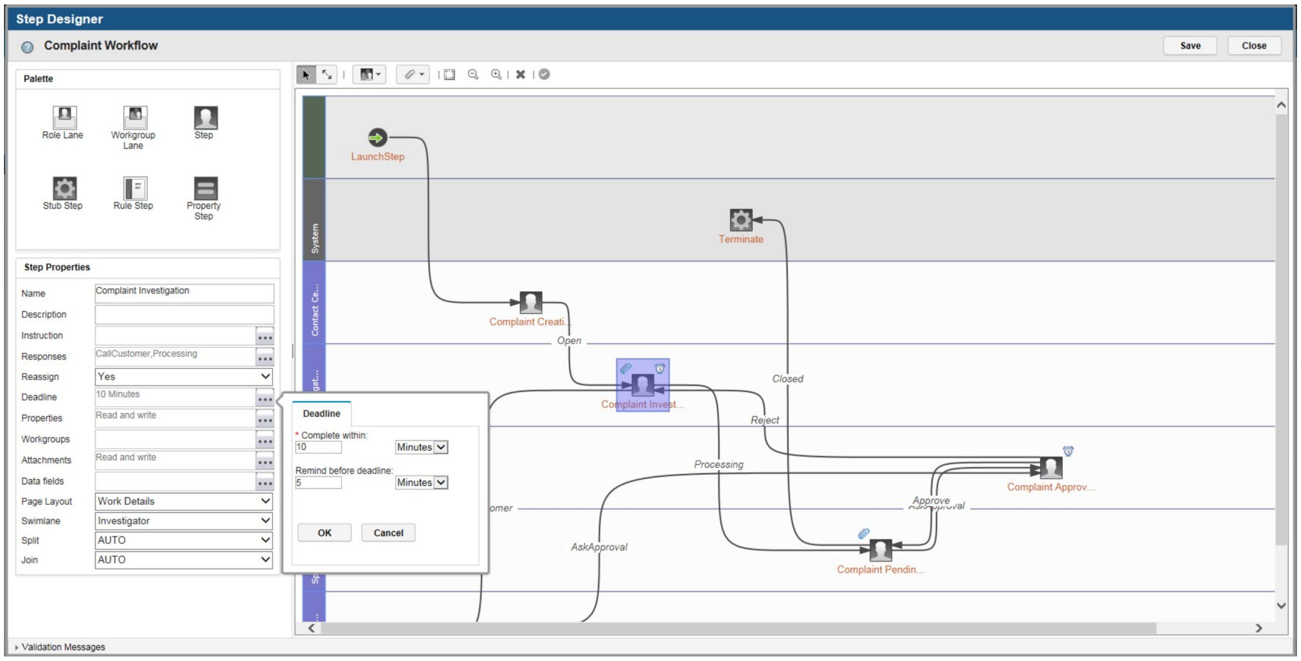
Customer Complaints: CC_ComplaintProcessing

How to define SLA

FileNet Process Engine allows to define SLA for every step in the process. Here is the article which describes such platform capability: [IBM Knowledge Center: Service-level agreements](#).

.

By default, it allows to set two amounts of time (in minutes, hours, days or weeks) for reminder (optional) and for the deadline of the step execution time frame. This is how ICM built-in Step designer allows to do this:



Step Properties

Name	Complaint Investigation
Description	
Instruction	
Responses	CallCustomer,Processing
Reassign	Yes
Deadline	10 Minutes
Properties	Read and write
Workgroups	
Attachments	Read and write
Data fields	
Page Layout	Work Details
Swimlane	Investigator
Split	AUTO
Join	AUTO

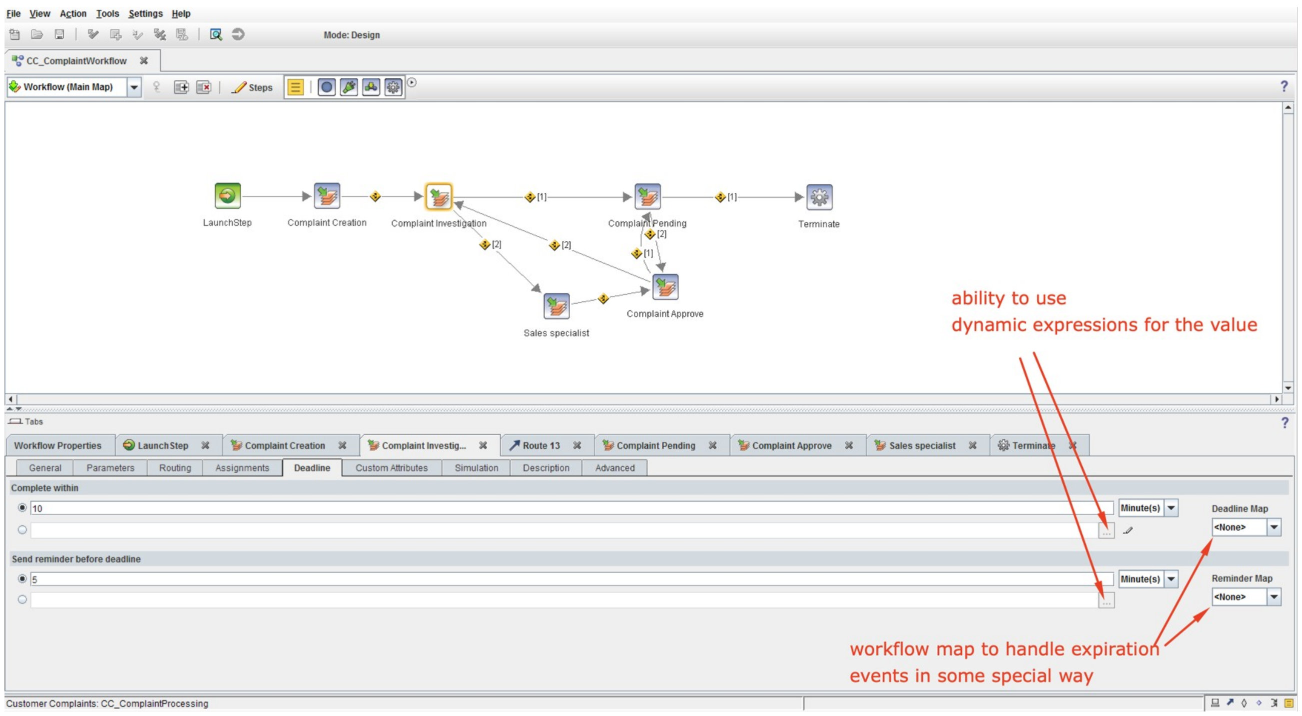
Deadline

* Complete within: 10 Minutes

Remind before deadline: 5 Minutes

OK Cancel

Process Designer applet, as an addition to the base capabilities of the Step designer, allows to use dynamic expressions, based on workflow, task or case field values. Also it allows to define custom workflow submap to handle both reminder and deadline expiration events in some special way.



Complete within
● 10 Minute(s)
Deadline Map
<None>

Send reminder before deadline
● 5 Minute(s)
Reminder Map
<None>

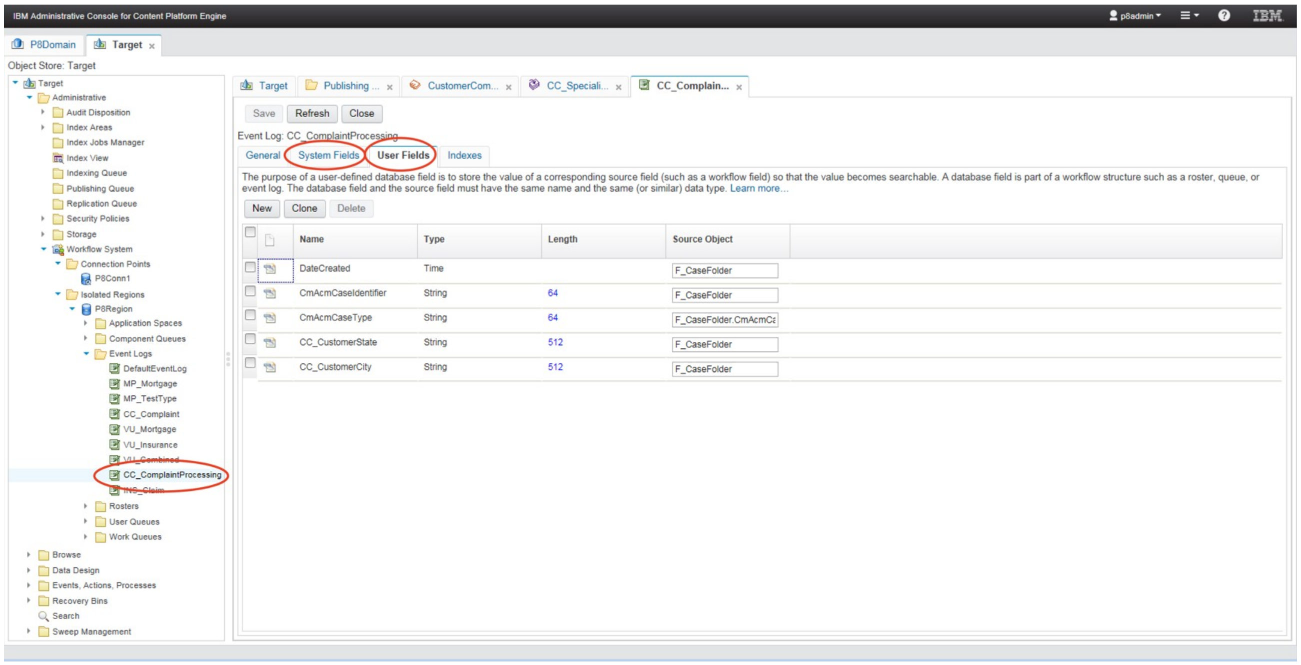
Also, Process Engine SLA capability allows to respect work schedules, see the following documentation article: IBM Knowledge Center: Work schedules.

Process Engine setup

It is required to prepare Process Engine entities before crawling. It needs to expose fields to roster, queues and event logs. All these actions can be performed using IBM Administrative Console for Content Platform Engine.

Preparing the Event log before crawling

In IBM Case Manager, every case type of the deployed solution brings the event log to store events of the workflows corresponding to case tasks.


Event Log: CC_ComplaintProcessing

The purpose of a user-defined database field is to store the value of a corresponding source field (such as a workflow field) so that the value becomes searchable. A database field is part of a workflow structure such as a roster, queue, or event log. The database field and the source field must have the same name and the same (or similar) data type. Learn more...

Name	Type	Length	Source Object
DateCreated	Time		F_CaseFolder
CmAcmCaseIdentifier	String	64	F_CaseFolder
CmAcmCaseType	String	64	F_CaseFolder.CmAcmCa
CC_CustomerState	String	512	F_CaseFolder
CC_CustomerCity	String	512	F_CaseFolder

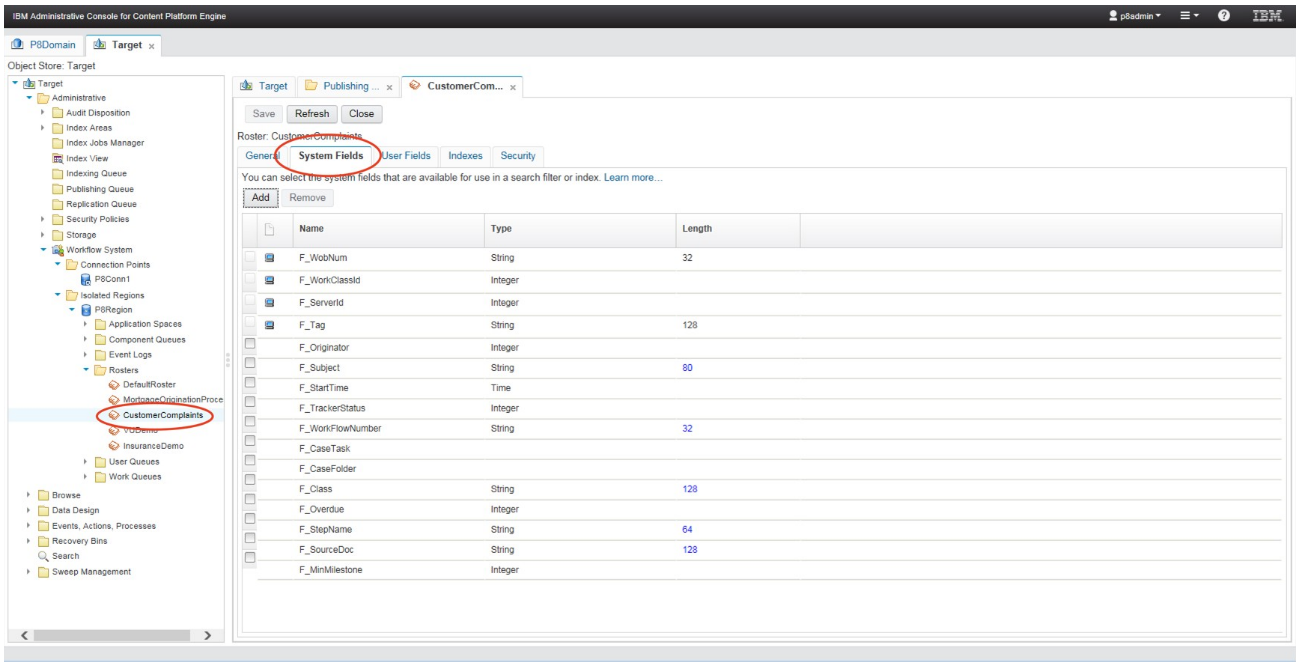
To prepare the event log to be crawled and analyzed do the following:

1. Add standard fields (“System Fields” tab): F_Class, F_StepName, F_TimeOut, F_Overdue
2. Add user defined fields (“User Fields”):

Field Name	Data type	Source object
DateCreated	Time	F_CaseFolder
CmAcmCaseType	String (64)	F_CaseFolder.CmAcmCaseTypeFolder.FolderName
CmAcmCaseIdentifier	String (64)	F_CaseFolder

Preparing roster and queues to display real-time data

In IBM Case Manager, every solution has its own roster. It needs to expose all default system fields to the roster. On the following screen click “Add” button and select all unexposed system fields.


IBM Administrative Console for Content Platform Engine

P8Domain | Target

Object Store: Target

- Target
  - Administrative
    - Audit Disposition
    - Index Areas
    - Index Jobs Manager
    - Index View
    - Indexing Queue
    - Publishing Queue
    - Replication Queue
    - Security Policies
    - Storage
    - Workflow System
      - Connection Points
        - P8Conn1
      - Isolated Regions
        - P8Region
          - Application Spaces
          - Component Queues
          - Event Logs
          - Rosters
            - DefaultRoster
            - MortgageOriginationProce
            - CustomerComplaints
            - VUDemo
            - InsuranceDemo
          - User Queues
          - Work Queues
  - Browse
  - Data Design
  - Events, Actions, Processes
  - Recovery Bins
  - Search
  - Sweep Management

Target | Publishing ... | CustomerCom...

Save | Refresh | Close

Roster: CustomerComplaints

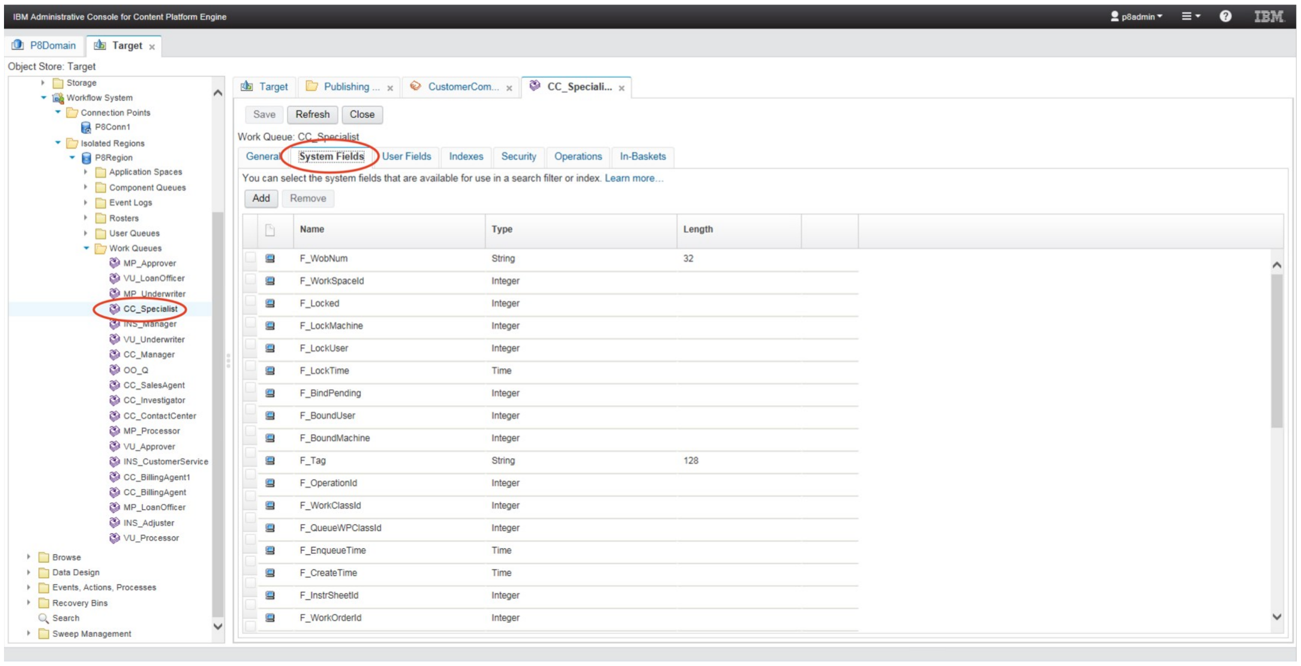
General | System Fields | User Fields | Indexes | Security

You can select the system fields that are available for use in a search filter or index. Learn more...

Add | Remove

Name	Type	Length
F_WobNum	String	32
F_WorkClassId	Integer	
F_ServerId	Integer	
F_Tag	String	128
F_Originator	Integer	
F_Subject	String	80
F_StartTime	Time	
F_TrackerStatus	Integer	
F_WorkFlowNumber	String	32
F_CaseTask		
F_CaseFolder		
F_Class	String	128
F_Overdue	Integer	
F_StepName	String	64
F_SourceDoc	String	128
F_MinMilestone	Integer	

Every user role in the ICM corresponds to the work queue. It is optional, depending on the dashboard requirements, but if it needs to show the queue elements in the Process Analytics feature, expose all available system fields exactly as it was done for the roster.


IBM Administrative Console for Content Platform Engine

P8Domain | Target

Object Store: Target

- Storage
- Workflow System
  - Connection Points
    - P8Conn1
  - Isolated Regions
    - P8Region
      - Application Spaces
      - Component Queues
      - Event Logs
      - Rosters
      - User Queues
      - Work Queues
        - MP_Approver
        - VU_LoanOfficer
        - MP_Underwriter
        - CC_Specialist
        - INS_Manager
        - VU_Underwriter
        - CC_Manager
        - OO_Q
        - CC_SalesAgent
        - CC_Investigator
        - CC_ContactCenter
        - MP_Processor
        - VU_Approver
        - INS_CustomerService
        - CC_BillingAgent1
        - CC_BillingAgent
        - MP_LoanOfficer
        - INS_Adjuster
        - VU_Processor
- Browse
- Data Design
- Events, Actions, Processes
- Recovery Bins
- Search
- Sweep Management

Target | Publishing ... | CustomerCom... | CC_Speciali...

Save | Refresh | Close

Work Queue: CC_Specialist

General | System Fields | User Fields | Indexes | Security | Operations | In-Baskets

You can select the system fields that are available for use in a search filter or index. Learn more...

Add | Remove

Name	Type	Length
F_WobNum	String	32
F_WorkSpaceId	Integer	
F_Locked	Integer	
F_LockMachine	Integer	
F_LockUser	Integer	
F_LockTime	Time	
F_BindPending	Integer	
F_BoundUser	Integer	
F_BoundMachine	Integer	
F_Tag	String	128
F_OperationId	Integer	
F_WorkClassId	Integer	
F_QueueWPClassId	Integer	
F_EnqueueTime	Time	
F_CreateTime	Time	
F_InstrSheetId	Integer	
F_WorkOrderId	Integer	

UIE crawler configuration

There are 3 plugins used to crawl both historical and real-time data from the ICM. To crawl the data configure PE entities as it was described in the previous section of this document.

Crawling aggregated historical data

IcmRepositoryPlugin

This plugin allows to crawl ICM process event log to extract the data about step executions, including historical information, SLA status, etc.

Sample configs are available with the distribution of the plugin. Configure connection using the following list of parameters: db connection string, schema name, user name and password. Adjust db views name to be crawled in StepExecutionsSQL and StepEventsSQL if needed (marked yellow in configuration example below): there are views for event logs.

IndexFields section (marked green in configuration example below) allows to configure fields to be added into the target index on crawling:

```
<IndexField name="$caseid_duplicate@s" expr="notNull('CmAcmCaseIdentifier@s').first().get('CmAcmCaseIdentifier@s')" />
```

Here: *name* – name of field in index, *expr* – expression to be applied to evaluate field value.

Configuration example:

```
<Repository ID="IcmRepository ">

    <ClassName>com.intellective.uie.plugin.repository.insights.icm.IcmRepository</ClassName>

    <Database>

        <Driver>com.ibm.db2.jcc.DB2Driver</Driver>

        <Type>db2</Type>

        <ConnectionString>jdbc:db2://server:50000/CASE_T</ConnectionString>

        <SchemaName>schemaName</SchemaName>

        <UserName>userName</UserName>

        <Password>password</Password>

    </Database>

    <RepositoryPlugins>

        <RepositoryPlugin ID="IcmPlugin_EventLog">

            <ClassName>com.intellective.uie.plugin.repository.insights.icm.IcmRepositoryPlugin</ClassName>

            <CrawlPlanID>CrawlerPlan_EventLog</CrawlPlanID>

            <ContentMimeTypes>

                <AcceptMimeTypes>

                    <MimeType>*</MimeType>

                </AcceptMimeTypes>

                <ErrorHandling>

                    ...

                </ErrorHandling>

            </ContentMimeTypes>
            <IndexFields>
                <IndexField name="$caseid_duplicate@s" expr="notNull('CmAcmCaseIdentifier@s').first().get('CmAcmCaseIdentifier@s')" />
                <IndexField name="$casecreated_duplicate@time" expr="notNull('DateCreated@time').first().get('DateCreated@time')" />
                <IndexField name="DateCreated@time" expr="toDateTime(notNull('DateCreated@l').last().get('DateCreated@l'))" />
                <IndexField name="DateCreated@d" expr="toDateTime(notNull('DateCreated@l').last().get('DateCreated@l'))" />
                <IndexField name="$queued_duplicate@time" expr="filter('F_EventType@i', 352).firstValue('F_TimeStamp@time')" />
                <IndexField name="$queued_duplicate_test@d" expr="filter('F_EventType@i', 352).firstValue('F_TimeStamp@time')" />
                <IndexField name="$customer_state@s" expr="notNull('CC_CustomerState@s').first().get('CC_CustomerState@s')" />
                <IndexField name="$customer_city@s" expr="notNull('CC_CustomerCity@s').first().get('CC_CustomerCity@s')" />
            </IndexFields>

            <Parameters>
                <TopN>10</TopN>
                <StepExecutionsSQL type="db2">
                    <![CDATA[
                        SELECT * FROM (SELECT "F_WobNum", "F_OccurrenceId", MAX("F_TimeStamp") AS MaxTime
                        FROM VWVL1_CC_COMPLAINTPROCESSING WHERE "F_EventType" IN (350,352,360,172,190,200) GROUP BY "F_WobNum", "F_OccurrenceId")
                        WHERE MAXTIME > :CRAWLPOINT_SEC OR ("F_OccurrenceId">:OCCURENCE_ID AND MAXTIME=:CRAWLPOINT_SEC)
                            ORDER BY MAXTIME, "F_OccurrenceId" fetch first 10 rows only
                        ]]>
                </StepExecutionsSQL>
                <StepEventsSQL>
                    <![CDATA[
                        SELECT * FROM VWVL1_CC_COMPLAINTPROCESSING WHERE "F_WobNum"=? AND "F_OccurrenceId"=? AND "F_EventType" IN
(350,352,360,172,190,200)
                            ORDER BY "F_TimeStamp"
                        ]]>
                </StepEventsSQL>
                <UserMappingSQL>
                    <![CDATA[
                        SELECT F_USERNAME AS UserName FROM VWUSER WHERE F_USERID=?
                        ]]>
                </UserMappingSQL>
            </Parameters>
        </RepositoryPlugin>
    </RepositoryPlugins>
</Repository>
```

Crawling real-time data

PE52RepositoryPlugin and CM52RepositoryPlugin are for crawling real-time data from the running processes and cases.

PE52RepositoryPlugin

This plugin is designed to crawl data from the running process instances, i.e. from the rosters and queues.

There are sample configs available with the distribution of the plugin. To configure repository connection data following parameters should be set to access FileNet

Process Engine via its API: BootstrapCEURI, connection point, user name and password. Adjust RosterName, EventLogName and user fields list if needed.

Configuration example:

```
<Repository ID="PE52Repository_CCPROCESSING" REFID="AbstractPERepository">

    <PE>

        <BootstrapCEURI>http://serverAddress:9080/wsi/FNCEWS40MTOM/</BootstrapCEURI>

        <ConnectionPoint>P8Conn1</ConnectionPoint>

        <UserName>userName</UserName>

        <Password>password</Password>

    </PE>

</Repository>

<Repository ID="AbstractPERepository">

    <ClassName>com.intellective.uie.plugin.repository.pe52.PE52Repository</ClassName>

    <RepositoryPlugins>

        <RepositoryPlugin ID="PE1">

            <ClassName>com.intellective.uie.plugin.repository.pe52.PE52RepositoryPlugin</ClassName>

            <CrawlPlanID>CM_PE_CrawlerPlan</CrawlPlanID>

            <Parameters>

                <TopN>100</TopN>

                <TimeLagSec>120</TimeLagSec>

                <RosterName>CustomerComplaints</RosterName>

                <EventLogName>CC_ComplaintProcessing</EventLogName>

                <UserIdFields>

                    <Field name="F_LockUser"/>

                    <Field name="F_BoundUser"/>

                    <Field name="F_Originator"/>

                </UserIdFields>

            </Parameters>

        </RepositoryPlugin>

    </RepositoryPlugins>

</Repository>
```

CM52Repository

This plugin allows to crawl objects from the FileNet object stores. For the context, it’s been used to crawl the data about cases and tasks objects.

Sample configs are available with the plugin distribution. Configure repository connection data: db connection string, schema name, user name and password.

Adjust list of ClassDefinition.SYMBOLIC_NAME(s) to be crawled if needed (marked yellow in configuration example below).

Configuration example:

```
<Repository ID="CMRepository" REFID="AbstractCM52Repository">
```

```
    <Database>

        <Driver>com.ibm.db2.jcc.DB2Driver</Driver>

        <Type>db2</Type>

        <ConnectionString>jdbc:db2://server:50000/CASE_T</ConnectionString>

        <SchemaName>schemaName</SchemaName>

        <UserName>userName</UserName>

        <Password>password</Password>

    </Database>

</Repository>

<Repository ID="AbstractCM52Repository">

    <ClassName>com.intellective.uie.plugin.repository.cm52.CM52Repository</ClassName>

    <RepositoryPlugins>

        <RepositoryPlugin ID="CM52">

            <ClassName>com.intellective.uie.plugin.repository.cm52.CM52RepositoryPlugin</ClassName>

            <CrawlPlanID>CM_PE_CrawlerPlan</CrawlPlanID>

            <Parameters>

                <TopN>1000</TopN>

                <TimeLagSec>120</TimeLagSec>

                <GetPropertiesSQL type="db2">

                    <![CDATA[

                select gpd.symbolic_name as property_name, td.table_name, cd.column_name, pd.datatype, gpd.cardinality,

                    pd.parent_id as class_id, cad.symbolic_name as class_name, pd.name_property_bool as is_title, pd.primary_id as property_id

                        from ::SCHEMA.PropertyDefinition pd

                        left join ::SCHEMA.ColumnDefinition cd on pd.column_id=cd.object_id

                        left join ::SCHEMA.TableDefinition td on pd.table_id=td.object_id

                        left join ::SCHEMA.GlobalPropertyDef gpd on pd.global_prop_id=gpd.object_id

                        left join ::SCHEMA.ClassDefinition cad on pd.parent_id=cad.object_id

                    where not cd.column_name is null

                    order by cad.symbolic_name]]>

                </GetPropertiesSQL>

                <GetMultiValueProperty>

                    <![CDATA[

                select parent_id as object_id, element_value as value, parent_prop_id as property_id

                        from ::SCHEMA.::TABLE_NAME where parent_id in (:DOC_IDS) order by parent_id]]>

                </GetMultiValueProperty>

                <JobSQL type="db2">

                    <![CDATA[select dv.*, (dv.modify_date-dv.create_date) AS case_lifetime from ::SCHEMA.Container dv

                inner join ::SCHEMA.ClassDefinition cad on dv.object_class_id=cad.object_id and cad.is_hidden=0 and sys_owned_bool=0

                where cad.SYMBOLIC_NAME IN ('CC_Complaint','CC_ComplaintProcessing') AND dv.CREATE_DATE > TO_DATE('2019-07-02 12-00-00', 'YYYY-MM-
DD HH24-MI-SS') AND (dv.modify_date > :CRAWLPOINT OR (dv.modify_date = :CRAWLPOINT AND dv.object_id > :CRAWLPOINT_OID))

                order by dv.modify_date asc, dv.object_id asc fetch first ::TOP_N rows only]]>

                </JobSQL>

                <LinkSQL>

                    <![CDATA[select * from ::SCHEMA.link lnk where object_class_id in (select cad.object_id from ::SCHEMA.ClassDefinition cad
where symbolic_name = 'CmAcmCaseRelationship')

                and lnk.head_id in (:CASE_OBJECT_IDS)]]>

                </LinkSQL>
                <AnnotationSQL>

                    <![CDATA[select * from ::SCHEMA.annotation where annotated_id in (:CASE_OBJECT_IDS)]]>

                </AnnotationSQL>

                <TaskSQL>

                    <![CDATA[select tsk.*, lnk.head_id from ::SCHEMA.task tsk inner join ::SCHEMA.link lnk on tsk.coordinator_id = lnk.tail_id
where lnk.head_id in (:CASE_OBJECT_IDS)]]>

                </TaskSQL>

                <DocumentsCheckSQL type="db2">

                    <![CDATA[select dv.*

                from ::SCHEMA.Container dv

                inner join ::SCHEMA.ClassDefinition cad on dv.object_class_id=cad.object_id and cad.is_hidden=0 and sys_owned_bool=0

                where ((dv.modify_date > :CRAWLPOINT and dv.modify_date < :CRAWLPOINT_MAX)

                 OR (dv.modify_date = :CRAWLPOINT AND dv.modify_date <> :CRAWLPOINT_MAX AND dv.object_id > :CRAWLPOINT_OID)

                 OR (dv.modify_date = :CRAWLPOINT_MAX AND dv.modify_date <> :CRAWLPOINT AND dv.object_id <= :CRAWLPOINT_OID_MAX)

                 OR (dv.modify_date = :CRAWLPOINT_MAX AND dv.modify_date = :CRAWLPOINT AND dv.object_id > :CRAWLPOINT_OID

                     AND dv.object_id <= :CRAWLPOINT_OID_MAX))

                order by dv.modify_date asc, dv.object_id asc fetch first ::TOP_N rows only]]>

                </DocumentsCheckSQL>

            </Parameters>

        </RepositoryPlugin>

    </RepositoryPlugins>

</Repository>
```

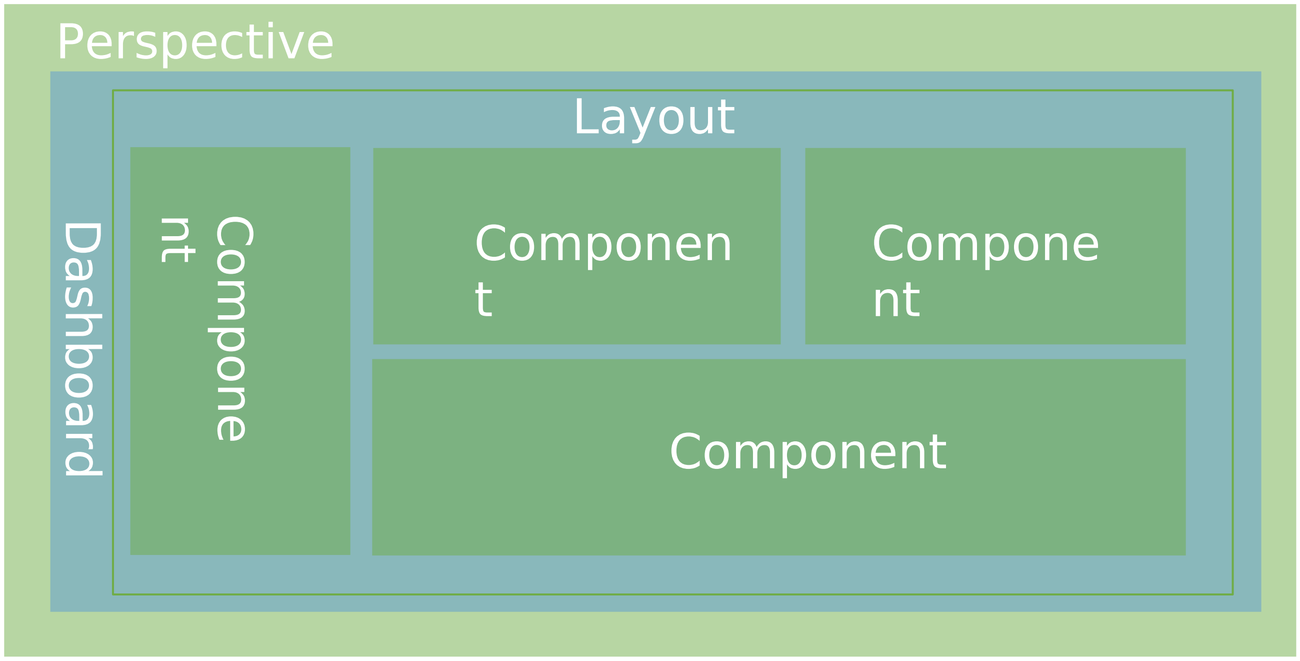
Unity Analytics

Unity Analytics features a few fully configurable and customizable analytics components that allow to visualize various kind of analytics data like aggregations, metrics and statistics:

- Field statistics
- Field metrics
- Charts: Line, Bar, Pie, Area, Composite, Heatmap, Treemap, etc
- Indicators
- Filters

UI Concept

Dashboard play an important role, allowing to quickly explore data, view key metrics at a glance and tailor the view based on the needs.



By the concept, Dashboard is a container for various UI components. Dashboards are also grouped by Perspectives which work as containers for dashboards and could potentially host as many dashboards as needed to describe a specific business realm.

Perspectives and Dashboards are fully configurable and are described with the following sections in the Unity configuration file.

```
<Perspective id="analytics" title="Analytics perspective" default="true">
	<Dashboard
		id="TaskAnalytics"
		builder="default"
		default="true"
		title="Process Trends"
		tooltip="Discovers process trends and insights">
```

```
        ...
    </Dashboard>
</Perspective>
```

Perspective attributes:

- Id – perspective identifier
- Title – perspective title
- Default flag – perspective is default and will be rendered by default, if set to true.
- Builder – assumes the existing builder component which is to be used for building and rendering a perspective. Default builder will be used if the attribute is empty.

Dashboard attributes:

- Id – dashboard identifier
- Title – dashboard title
- Tooltip – dashboard tooltip
- Builder - assumes the existing builder component which is to be used for building and rendering the dashboard. Default builder will be used if the attribute is empty.
- Lazy flag – dashboard will be only loaded upon explicit request, if set to true. Flag is set to true, by default.

Dashboard section includes UI component references.

```
<Perspective default="true" id="analytics" title="Analytics perspective">
    <Dashboard default="true" id="TaskAnalytics" title="Process Trends">
        <Component ref="step-duration-avg" type="indicator" />
        ...
    </Dashboard>
    ...
</Perspective>
```