Appway

Appway Business Activity Monitoring (BAM)

Issued - 30.09.2013

Appway AG
Lessingstrasse 5, CH-8002 Zürich, Switzerland
developer@appway.com
www.appway.com

Appway

Table of Contents

1 Basic Concepts 5
1.1 Definition 5
1.2 SLAs and Scenarios 5
1.3 Business Indicators Monitoring 6
1.4 Dashboards 7
1.5 Usage Steps 7
2 Installation and Configuration 9
2.1 Installation Procedure 9
2.2 Additional Configuration 12
3 Architecture 14
3.1 BAM Modules 14
3.1.1 SLA Tool 14
3.1.2 Process Listener 14
3.1.3 Extraction Functions 15
3.1.4 Out-of-the-box Dashboards 15
3.2 Database Structure 15
3.2.1 SLA Tables 15
3.2.2 Runtime Data Tables 17
3.3 Functions 21
3.3.1 Utility Functions 22
3.3.2 Process Functions 23
3.3.3 Extraction Functions 24
4 Studio Modules 29
4.1 Service Level Agreements 29
4.1.1 Overview of Processes 29
4.1.2 SLA Editor 30
4.2 BAM Dashboard 32
4.3 BAM Test Data Generator 32
4.4 SLA Report 33
5 Additional Components 35
5.1 Portal Worklist SLA Columns 35
6 Out-of-the-Box Dashboards 37
6.1 Prerequisites 37
6.2 Installation 37

6.3 Upgrade 37
6.4 Filters 38
6.5 Dashboard Tabs 40
6.5.1 Performance Metrics 40
6.5.2 Instances Overview 41
6.5.3 Activities Overview 42
6.5.4 User Activities Overview 43
6.6 Business Attributes Configuration 44
6.7 Trackpoints 45
6.8 Reports 46
6.8.1 Instance Report 46
6.8.2 Activities Report 46

7 Dashboard Customization 48
7.1 Screen Structure and Templates 48
7.2 Supported Customization Types 50
7.3 Customization Guidelines 52
7.3.1 Dashboard Top Header 52
7.3.2 Dashboard CSS 52
7.3.3 New Performance Metrics Charts 53
7.3.4 Remove Out-of-the-Box Performance Metrics Charts 55
7.3.5 New Dashboard Tab 55
7.3.6 Remove Out-of-the-Box Tabs 57
7.3.7 Custom Actions Column in the “Instances Overview” Tab 57
7.3.8 Custom Actions Column in the “Activity Details” Screen 59
7.4 Customization Examples 60

This document applies to the Appway BAM extension, version 1.7.1.1

1 Basic Concepts

1.1 Definition

According to Wikipedia's definition, Business activity monitoring (BAM) is software that aids in the monitoring of business activities, those activities being implemented in computer systems.

The goals of business activity monitoring are to provide real-time information about the status and results of various operations, processes and transactions. The main benefits of BAM are to enable an enterprise to make better-informed business decisions, quickly address problem areas, and re-position itself to take full advantage of emerging opportunities.

The term was originally coined by analysts at Gartner Inc. and refers to the aggregation, analysis and presentation of real-time information about activities inside organizations, and those involving customers and partners. A business activity is a business process, either orchestrated by business process management (BPM) software, or consisting of a series of activities spanning multiple systems and applications. BAM is an enterprise solution primarily intended to provide a real-time summary of business activities to operations managers and upper management.

In line with this definition, the purpose of the Appway BAM component is to provide managers with real-time visibility and process control over Appway instances, allowing them to monitor business data and events, and to initiate appropriate actions.

In practical terms, by looking at BAM dashboards, managers will be able to monitor and measure processes, identify optimization potential and redesign the processes accordingly if necessary.

1.2 SLAs and Scenarios

The monitoring is essentially based on time measurements of user interactions and activities. BAM allows the user to define expected durations for processes and activities, and running processes are measured against these predefined durations in order to identify bottlenecks and optimization points.

These expected durations are called **Service Level Agreements (SLA).** They can be considered as “contracts” between the service requestor and the service provider, defining the maximum expected/allowed time for the completion of activities.

Both the managers and the users will have visibility into the SLA status of the tasks. Users will use this indication to react appropriately on urgent tasks, and managers will use it to monitor the overall status.

The action to be taken on expiration of an activity is to be defined in each specific case. Examples actions include sending out notifications emails to interested parties, applying reassignment rules to redistribute the workload, and putting in place escalation mechanisms for immediate reaction. No predefined action or rule is specified inside BAM: its role is to identify and mark the activities and processes that are expired.

The SLAs can be specified at two different levels: process SLAs and activity SLAs. A **Process SLA** is the expected duration of the overall process, e.g. client onboarding or KYC. An **Activity SLA** is the expected duration of one sub-activity of the process, e.g. entering client information.

BAM does not force the SLA of a process to be the sum of its sub-activities. Therefore, a "buffer" SLA can be considered on the process duration. Also, the SLA status of an activity is not necessarily the same as the SLA status of the process containing it. For example, there are cases where the activity itself is on time, but the overall process is already overdue. On the other hand, the single activity may be overdue, but the overall process has just started and is therefore still on time.

For each activity and process SLA, it also possible to specify a tolerance **threshold.** This value, provided as a percentage of the SLA, indicates when the activity should produce a warning. If a threshold is specified, the user is able to react to the warnings and take action before it is too late, meaning before the SLA is expired.

There are cases when the same activity or process could have a different expected duration depending on the circumstances. For example, opening a joint account could take longer than opening an individual one, as multiple parties are involved. In this case, it is important to be able to specify a different SLA for the process and its activities in the two different cases, i.e. making the SLA durations dependent on process variations.

Appway BAM uses the concept of **scenario** to implement this requirement. A scenario is a defined set of process instance characteristics which determines a specific execution path in the flow. In the example below, the type of account defines the flow inside the account opening process.

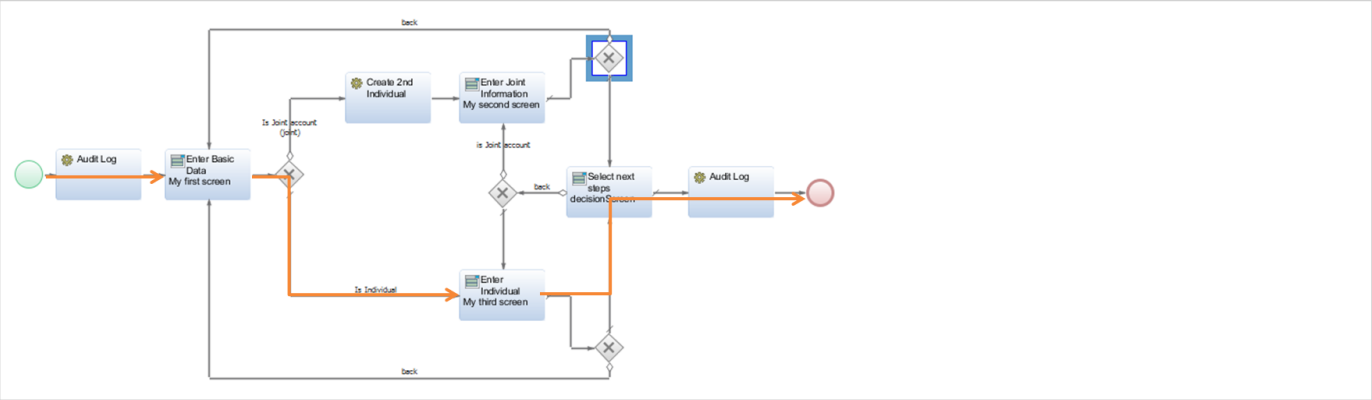


fig 1: Example of scenarios inside an account opening process

BAM allows the user to specify different scenarios for a process, and different process SLA and activity SLAs for each scenario defined. Also, it is possible to select a default scenario for a process. All new instances of that process will be created under the default scenario selected. The scenario can be easily set/changed in a process instance by invoking a script function provided by the BAM extension (SetProcessSLAScenario). After the scenario is set, the SLA status of the process instance is updated to match the new scenario SLA. All sub-activities started after the change use the new activity SLAs.

1.3 Business Indicators Monitoring

In parallel to the standard SLA-based time monitoring, BAM provides monitoring capabilities based on custom business indicators. It might be relevant to monitor and filter specific processes based on the region or market they refer to. It might also be relevant to track a certain money amount being recorded in the process in order to compute trends and aggregate values.

The mechanism that BAM offers to save these indicators is called **trackpoints.** Trackpoints are out-of-the-box components that can be used in the process modeler to record a relevant business attribute of the process. They are visually marked in the flow in order to easily identify points of the process where the recording happens. Behind the

scenes, the trackpoints are saved as process instance attributes. The values can then be used for filtering and displaying purposes in the dashboards.

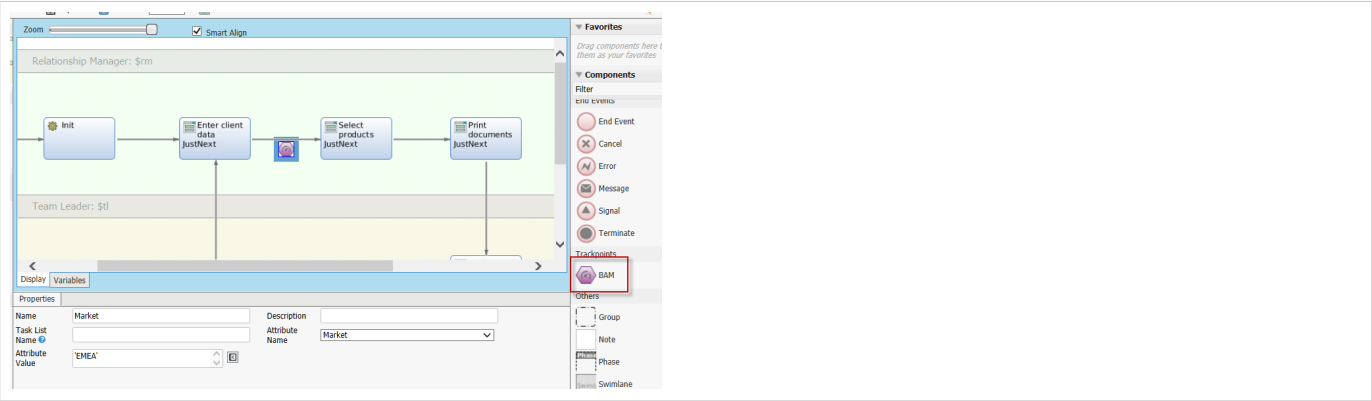


fig 2: A BAM trackpoint in the process modeler

1.4 Dashboards

The Appway BAM offers out-of-the-box dashboards allowing the business user to monitor the status of instances, activities and SLA, aggregate processes and activity indicators, and report on the available data via MS Excel. The dashboards also offer a drill-down feature from the general overview to specific lists of filtered processes down to single instances and their sub-activities. Advanced data filtering capabilities are provided in order to quickly identify the relevant subset of instances and activities for aggregation or reporting purposes. Finally, a set of predefined clickable charts are available as a general overview of the overall system status.

The basic out–of-the-box dashboards come with the BAM extension and are implemented using Appway screens and processes. They are deployed to the environment using the standard content import feature. As a consequence, the dashboards can be fully customized according to the specific project needs. Customizations include graphical and layout adjustments, implementation of additional charts and modification of the existing ones, extension of the report tables and basically any other change that is supported by Appway screen design.

1.5 Usage Steps

The standard BAM usage scenario includes four steps. The first one is **defining the business process.** This is the standard Appway business process design enhanced with BAM-specific features like the trackpoints.

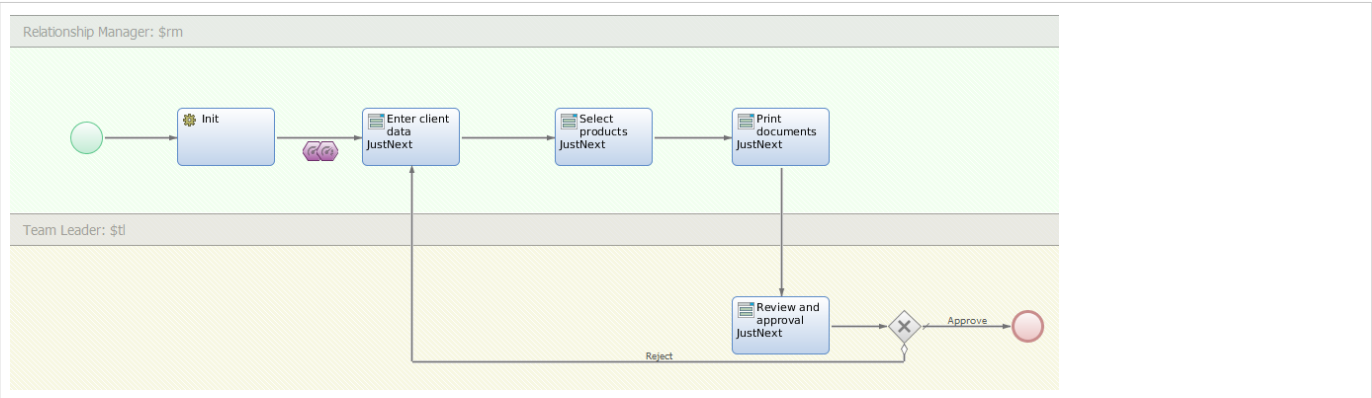


fig 3: Example definition of a business process

The second step is **defining the SLAs** on the process, which includes creating all necessary scenarios and setting SLAs

on each relevant activity. This step is mandatory, as only activities with an SLA are saved to BAM. In other words, when BAM is installed by default, no data is stored yet. Only after the definition of one or more SLAs does BAM start saving the related runtime data.

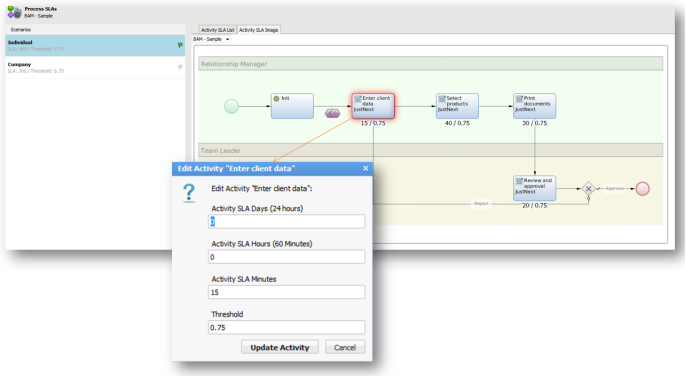


fig 4: Definition of the SLAs on the business process

The third step is **running the process** as usual in the workspace. BAM will take care of storing all runtime process data including status, duration and custom business indicators.

The last step is **monitoring the status of processes** in the BAM dashboards. The dashboards provide a rich interface for monitoring, filtering and reporting.

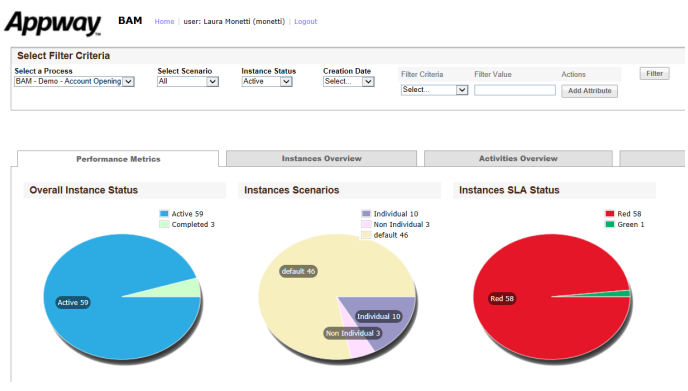


fig 5: Process monitoring using the BAM dashboards

2 Installation and Configuration

2.1 Installation Procedure

The creation of a database and its setup in Appway is a prerequisite for the installation of BAM.

The database management systems (DBMS) supported are H2 (testing performed on version 1.3) and Oracle (10g and 11g).

H2 can only be used for prototype and showcase environments such as proofs of concept. Please follow the instructions on Data Sources (http://developer.appway.com/screen/ViewDocument?bookId=1272417269542) in order to setup an H2 database in Appway.

The only currently supported production database is Oracle. Before starting with the BAM setup in Appway, it is necessary to create an Oracle instance (an existing one can be used) and a schema (dedicated to BAM). The database user must have connect, select, insert and update rights, table and index creation rights plus procedure creation and invocation rights and unlimited tablespace quota.

Once the schema is created, it is necessary to set up an Appway Data Source that points to the newly created database. Instructions on setting up an Oracle Data Source can also be found here (http://developer.appway.com/screen/ViewDocument?bookId=1272417269542) .

The name of the Appway Data Source for H2 and Oracle that will be used for BAM is not relevant and can be any valid data source name.

The BAM is delivered as an Appway extension. After the Data Source creation, the BAM extension can be uploaded to the Appway environment and started. Please note that BAM requires Appway release 5.3.

Upon every BAM extension restart, Appway will notify that the Studio must be reloaded. This step is mandatory for a correct functioning of the BAM modules in the Studio.

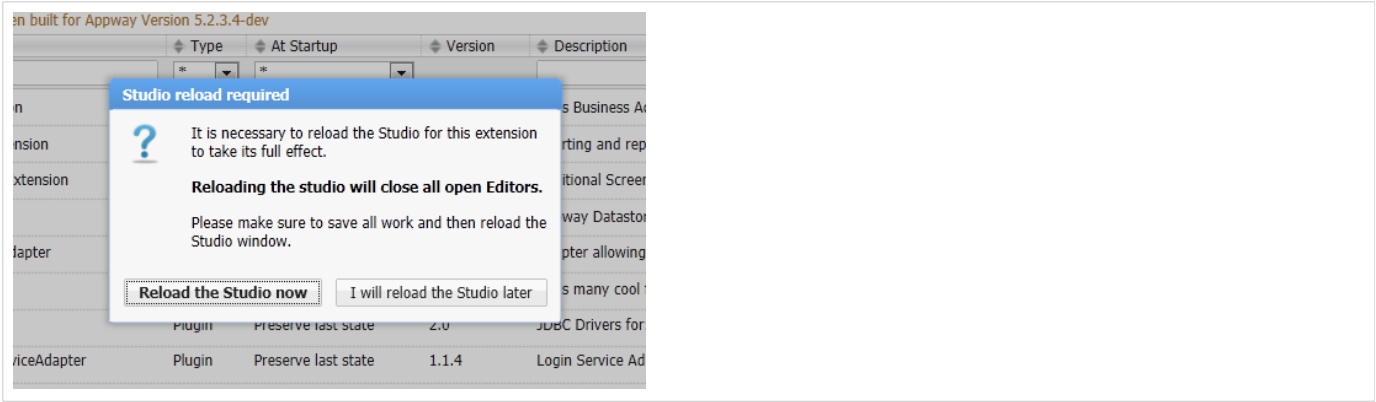


fig 6: After the BAM extension is installed, a warning requires confirmation for Studio reload

After the BAM extension's startup, a new "BAMExtension" section will appear under Administration. This contains a configuration module, "BAM Installation", that allows the user to set up and manage the database content.

The first step for the configuration of BAM is the selection of the data source name and type.

In the top section of the BAM Installation page, select the data source name and its correct type (H2/Oracle). Afterwards, click the **"Update"** button to update the configuration.

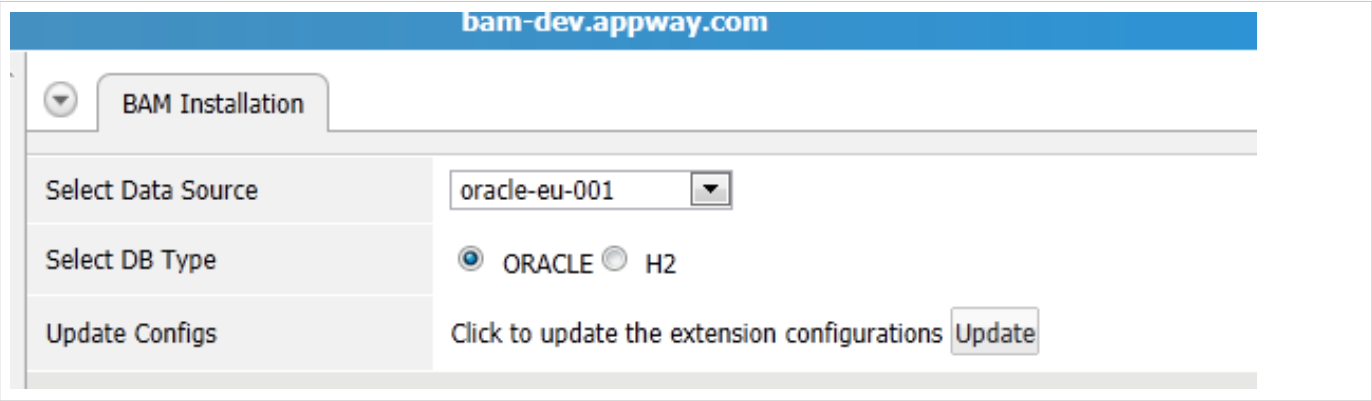


fig 7: Database configuration in the BAM Installation page

Immediately after the data source selection, Appway will check if the BAM database tables are present. During the first configuration, these will of course be missing; therefore Appway will show the corresponding warnings.

To solve this, the next step required is setting up all the database tables and indexes that are used to store BAM data. Click on the **"Create"** button in the "Actions" section.

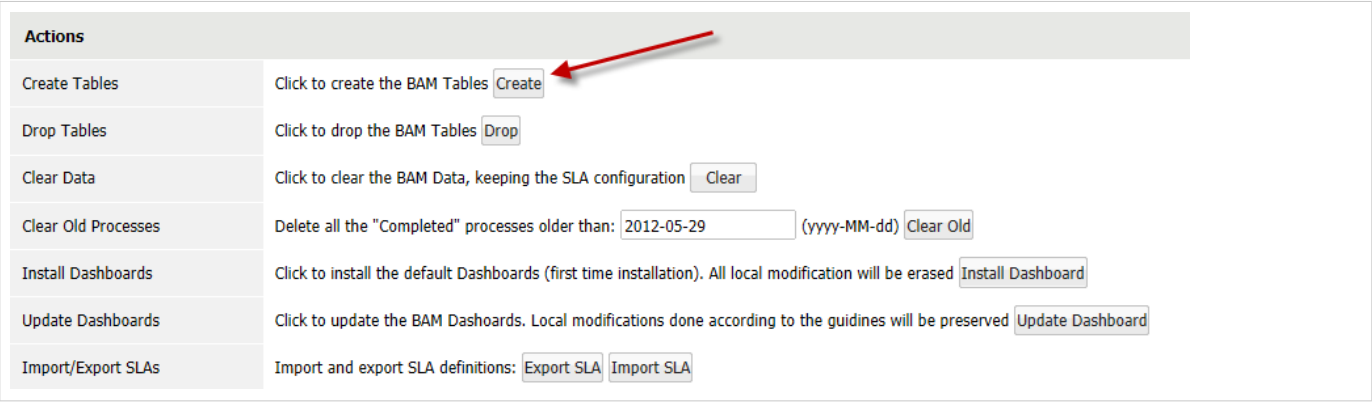


fig 8: Actions available in the BAM Installation page

If any problem occurs during this step, the tables can be dropped in order to have a clean situation by clicking on the "Drop" button. Please be extremely careful in using this in production environments, as this will drop all the tables and remove all the data from the database.

The "Actions" section also presents some additional features (non-mandatory during installation):

Button	Description
Clear	Deletes all BAM data. This will preserve the table definition and the SLA values.
Clear Old	Deletes of all the Completed and Cancelled process instances older than a specified date. This will preserve the table definition and the SLA values.

Install Dashboards	Imports the out-of-the-box BAM dashboards shipped with the extension. The button will open the standard import interface in order to check the result of the import and resolve any conflict. This button is used for the first installation of the out-of-the-box dashboards; it imports all content including the dashboard templates and the default settings. The "Out-of-the-box Dashboards" section of this document provides more information about this step and its prerequisites.
Update Dashboards	Updates the out-of-the-box BAM dashboards shipped with the extension. The button will open the standard import interface in order to check the result of the import and resolve any conflict. This button is used when updating the out-of-the-box dashboards to the latest version available after having installed the new version of the BAM extension. It imports the dashboard templates only and it preserves local modifications to the settings done according to the guidelines. The "Out-of-the-box Dashboards" section of this document provides more information on this step.

The Import/Export SLA buttons allow the user to export all SLA definitions from an environment and then import them in bulk into another instance, for example when moving from the development (DEV) to the production (PROD) environment.

Button	Descprition
Export SLA	Generates an .sla file and allows the user to save it to the local drive.
Import SLA	Lets the user select an existing .sla file from the local drive and imports it into the environment. The import operation will override all SLA definitions related to processes that are contained in the .sla file.

Finally, the last section of the installation page presents a summary of the BAM tables with the number of records for each. The first 4 tables contain runtime data of processes and activities, the last 2 tables contain SLA values for processes and activities.



Status

Table	Row Count
Workflow Instances	60
Workflow Instance Attributes	295
Workflow Tokens	752
Workflow Tokens Attributes	6
Process SLA	9
Activity SLA	54

fig 9: Database status with counters available in the BAM Installation page

2.2 Additional Configuration

Some additional configuration parameters can be specified in the extension configuration. These include:

Parameter	Default value	Description
worklist.attributes	*	Lets the user specify which process instance attributes should be considered BAM-relevant, and therefore saved in the BAM database. By default all attributes are saved (worklist.attributes =*). If this property contains a coma-separated list of attribute names, then only those attributes are saved.
bam.ignoreAttributes	empty	Lets the user specify which process instance attributes are not BAM-relevant. This property contains a coma-separated list of attributes that should be ignored. The following attributes are always ignored: processExpirationTime, processWarningTime, activityExpirationTime, activityWarningTime, BAM_Scenario.
bam.log.level	ERROR	Lets the user specify the log level for all the BAM extension classes at extension start-up.
bam.screen.task.enabled	true	Enables/disables the recording of screen tasks in BAM.

bam.intermediate.events.enabled	false	Enables/disables the recording of intermediate events in BAM (timer, receive message and receive signal intermediate events).
bam.listener.priority	95	Lets the user specify the priority used when triggering the process listener in order to coordinate the interaction with other extensions such as the Offline Worklist.
bam.testDataEnabled	false	Enables the section BAM Test Data Generator under the Monitor section of the Appway studio.

3 Architecture

3.1 BAM Modules

The Appway BAM is composed of a set of different modules: an SLA tool, a process listener, extraction functions, out-of-the-box dashboards, and database storage (BAM DB). The first four modules are described below; for the BAM DB module, see section 3.2 on the database structure.

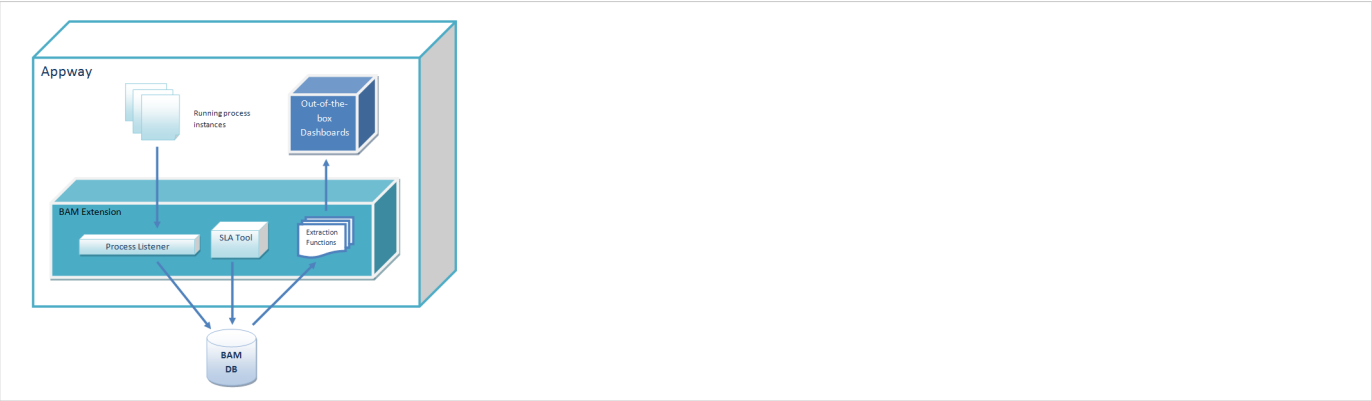


fig 10: High-level description of the interaction and data flow among the components

3.1.1 SLA Tool

The SLA tool is the first access point for BAM. The user is required to set the SLA values for all relevant processes and activities, which are saved in the BAM DB in dedicated tables. The table "BAM_ProcessSLA" contains the SLA values for processes, and the table "BAM_ActivitySLA" contains the SLA values for activities.

3.1.2 Process Listener

The second component that comes into play is the process listener. The process listener is able to intercept events happening in the Appway process engine and related to running process instances. This includes events like starting a new process instance, reassigning a process instance, setting an attribute on a process instance, completing a process instance, etc.

The BAM process listener records these events in the BAM DB so that it is then possible to run queries against this runtime data and display the results in the dashboards. It is important to remember that the listener only records data for processes and activities that have an SLA defined on them (different from 0). The other processes and activities are ignored.

In addition to recording this basic runtime data, the process listener is able to compute the expiration and warning time for each process and activity by accessing the SLA tables on the BAM DB. The expiration time is computed as starting time + SLA, while the warning time is computed as starting time + (SLA*threshold). These expiration and warning times are saved in the BAM DB together with the standard runtime data.

In addition, the listener also sets these times as process and token attributes on the processes and activities. The process attributes are called processExpirationTime and processWarningTime and the activity attributes are called activityExpirationTime and activityWarningTime. All values are saved as timestamps (Long numbers).

3.1.3 Extraction Functions

The next module to be taken into account consists of extraction functions. This set of Appway functions can be used in the Appway scripting language and execute standard extraction queries on the BAM DB. These functions are implemented in the BAM extension using direct SQL queries and Java code; therefore their code is not available/editable from the Appway studio.

Examples of these functions are computing the average instance duration by month, counting the number of instances by scenario and retrieving the warning time for an activity. The full list with details for each function will be provided in the next sections.

3.1.4 Out-of-the-box Dashboards

The last component consists of out-of-the-box dashboards. Unlike all the other components described above which are built in the extension source code, this module is built in the Appway Studio using standard Business Objects (screens, processes, labels, etc). This implies that it can be fully adapted and customized.

The dashboards invoke the extraction functions in order to retrieve data from the BAM DB and then display these using tables and charts.

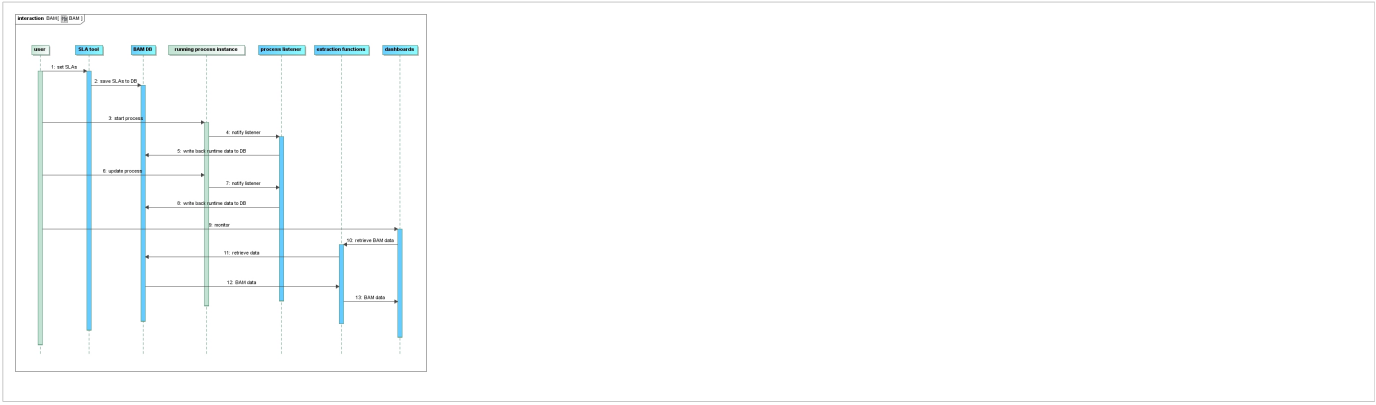


fig 11: Sequence diagram illustrating the interaction among the different actors in a typical Appway BAM usage scenario

3.2 Database Structure

The BAM DB includes tables that store runtime process and activity data, and SLA values. The schema includes 6 tables, 2 of them dedicated to SLAs and 4 dedicated to runtime data.

3.2.1 SLA Tables

The SLA tables are:

- "BAM_ProcessSLA": contains process SLAs and thresholds
- "BAM_ActivitySLA": contains activity SLAs and thresholds

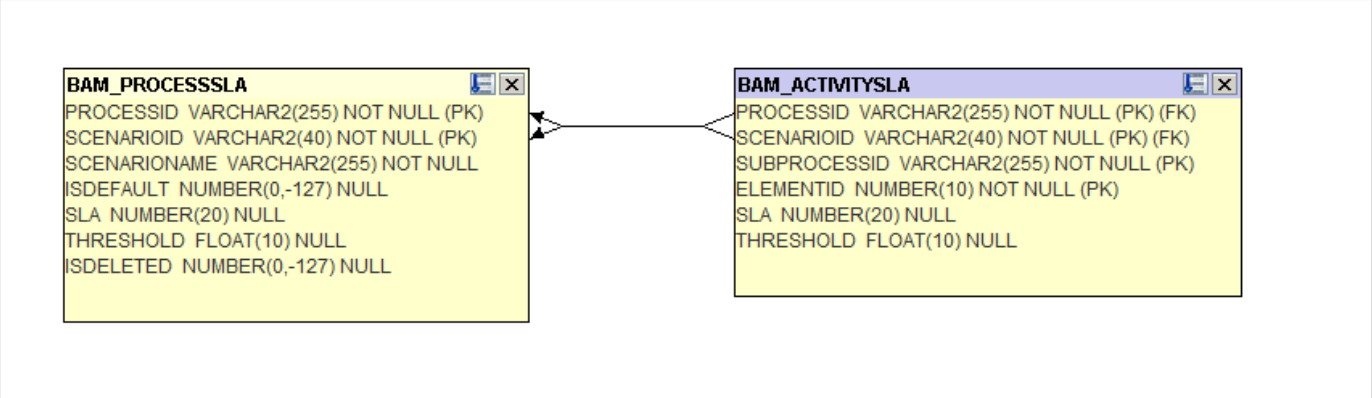


fig 12: Appway BAM database schema (SLA tables)

The **BAM_ProcessSLA** table contains one row for each process, scenario pair. The columns of this table are:

Column Name	Data Type	Description
processId	string	id of the process, e.g. 'AccountOpening'
scenarioId	string	id of the scenario, e.g. 'indiv'
scenarioName	string	name of the scenario, e.g. 'Individual'
isDefault	integer 0/1	flag to mark default scenario for a process
sla	number	SLA value in minutes
threshold	number between 0 and 1	threshold value
isDeleted	integer 0/1	flag to mark deleted scenarios

Table 4: BAM_ProcessSLA table

The primary key of the table is (processId,scenarioId).

The **BAM_ActivitySLA** table contains one row for each activity, process and scenario set. The columns of this table are:

Column Name	Data Type	Description
processId	string	id of the process, e.g. 'AccountOpening'
scenarioId	string	id of the scenario, e.g. 'indiv'
subProcessId	string	

		id of the subprocess where the activity is contained, e.g. 'SearchClient'
elementId	string	id of the activity inside the process / sub-process
sla	number	SLA value in minutes
threshold	number between 0 and 1	threshold value

Table 5: BAM_ActivitySLA table

The primary key of the table is (processId,scenarioId,subProcessId,elementId). subProcessId and elementId together identify an activity inside a process. There is a foreign key reference to the (processId,scenarioId) key in BAM_ProcessSLA.

3.2.2 Runtime Data Tables

The runtime data tables are:

- "BAM_WorkflowInstances": contains runtime data of process instances
- "BAM_WorkflowInstanceAttributes": contains process instance attributes
- "BAM_WorkflowTokens": contains runtime data of process tokens (activities)
- "BAM_WorkflowTokenAttributes": contains process token attributes

The definition includes appropriate database indexes for each table. No foreign key constraint is applied to these 4 tables.

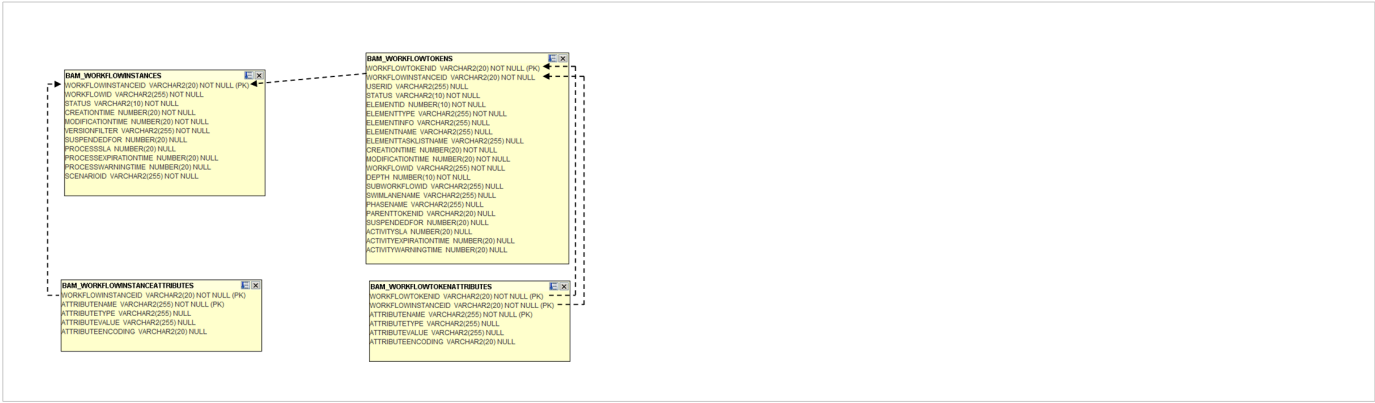


fig 13: Appway BAM database schema (runtime data tables)

The **BAM_WorkflowInstances** table contains one row for each "BAM-relevant" process instance (SLA defined and different from 0). The columns of this table are:

Column Name	Data Type	Description
workflowInstanceId	string	id of the process instance, e.g. '1234567890'
workflowId	string	

		id of the process, e.g. 'AccountOpening'
status	'Active'|'Cancelled'| 'Completed'|'Suspended'	process instance status
creationTime	timestamp number	creation time of the process instance
modificationTime	timestamp number	modification time of the process instance
versionFilter	string	version filter attached to the process instance, e.g. 'TimestampFilter-1366980809123', 'HeadFilter-permanent', etc.
suspendedFor	number in milliseconds	the total amount of time during which the process instance was in status "Suspended" (number in milliseconds)
processSLA	number in minutes	current SLA of the process, depending on the current scenario
processExpirationTime	timestamp number	expiration time for the process depending on the current SLA. This is computed as creationTime + SLA
processWarningTime	timestamp number	warning time for the process depending on the current SLA and threshold. This is computed as creationTime + SLA*threshold.
scenarioId	string	current scenarioId for the process instance, e.g. 'indiv'. If updated, the processSLA, processExpirationTime and processWarningTime are also updated.

Table 6: BAM_WorkflowInstances table

The primary key of the table is the workflowInstanceId.

The **BAM_WorkflowInstanceAttributes** table contains one row for each BAM-relevant process instance and instance attribute. The columns of this table are:

Column Name	Data Type	Description
workflowInstanceId	string	id of the process instance, e.g. '1234567890'

attributeName	string	name of the process token attribute, e.g. 'clientRegion'
attributeType	string	type of the attribute value, e.g. 'java.lang.String'
attributeValue	string	value of the attribute for the process token, e.g. 'EMEA'
attributeEncoding	string	encoding of the attribute value, if any

Table 7: BAM_WorkflowInstanceAttributes table

The primary key of the table is (workflowInstanceId,attributeName).

The **BAM_WorkflowTokens** table contains one row for each BAM-relevant process token (activity). The columns of this table are:

Column Name	Data Type	Description
workflowTokenId	string	id of the process token, e.g. '1234567890'
workflowInstanceId	string	id of the process instance, e.g. '1234567890'
userId	string	id of the current owner of the activity, e.g. 'nm'
status	'Active'|'Cancelled'| 'Completed'|'Suspended'	process token status
elementId	string	id of the activity inside the process / sub-process
elementType	'SubWorkflow'|'ScreenTask'| 'IntermediateEvent'	type of the activity
elementInfo	string	detail information used only for intermediate events, e.g. signal name, message queue, etc.
elementName	string	name of the activity inside the process / sub-process
elementTaskListName	string	tasklist name of the activity inside the process / sub-process
creationTime	timestamp number	process token's creation time

modificationTime	timestamp number	process token's modification time
workflowId	string	id of the process, e.g. 'AccountOpening', This is the id of the lowest level process (in case the token is on a subprocess) and does not necessarily match with the workflowId column of the BAM_WorkflowInstances table which always contains the top level process id.
depth	number	depth level in the sub-process hierarchy for the current token, e.g. 3
subWorkflowId	process id	if the token resides on a sub-process, this contains the id of the sub-process inside the container process
swimlaneName	string	current swimlane name for the process token. This travels up the sub-process hierarchy to find the lowest level swimlane available, e.g. 'RiskTeam'
phaseName	string	current phase name for the process token. This travels up the sub-process hierarchy to find the lowest level phase available, e.g. 'Identify Client'.
parentTokenId	string	lowest level, BAM-relevant parent token available for the current token. This travels up the sub-process hierarchy to find the lowest level parent token that is available in the BAM_WorkflowTokens table. This is different from the standard parent token if the standard one does not have an SLA defined on it and is therefore not saved in BAM DB.
suspendedFor	number in milliseconds	total amount of time during which the process token was in status 'Suspended'
activitySLA	number in minutes	SLA of the activity, depending on the scenario at the moment of creation (number in minutes). This value is never updated for an existing activity.
activityExpirationTime	timestamp number	

		expiration time of the activity depending on the SLA. This is computed as creationTime + SLA. This value is never updated for an existing activity.
activityWarningTime	timestamp number	warning time of the activity depending on the SLA and threshold. This is computed as creationTime + SLA*threshold. This value is never updated for an existing activity.

Table 8: BAM_WorkflowTokens table

The primary key of the table is the workflowTokenId.

The **BAM_WorkflowTokenAttributes** table contains one row for each BAM-relevant process token and token attribute. The columns of this table are:

Column Name	Data Type	Description
workflowTokenId	string	id of the process token, e.g. '1234567890'
workflowInstanceId	string	id of the process instance, e.g. '1234567890'
attributeName	string	name of the process token attribute, e.g. 'clientRegion'
attributeType	string	type of the attribute value, e.g. 'java.lang.String'
attributeValue	string	value of the attribute for the process token, e.g. 'EMEA'
attributeEncoding	string	encoding of the attribute value, if any

Table 9: BAM_WorkflowTokenAttributes table

The primary key of the table is (workflowTokenId,workflowInstanceId,attributeName).

3.3 Functions

The Appway BAM provides a set of out-of-the-box extraction and utility functions. As mentioned, these functions are implemented in the BAM extension using direct SQL queries and Java code; therefore their code is not available/editable from the Appway Studio. This chapter provides a list of these functions organized in sets, with a description of each.

3.3.1 Utility Functions

Function	Description
GetBAMProcesses	Returns a list of all processes that are BAM relevant, i.e. which have at least one scenario with at least one SLA. The result is a named collection; its keys are process ids and values are process names (e.g. {'TestProcess'= 'Test Process', 'AOP'= 'Account Opening'}:String).
GetProcessScenarios	Returns a list of defined scenarios given the process id. The result is a named collection; its keys are scenario ids and whose values are scenario names (e.g. {'1'= 'default', 'test'= 'Test Scenario'}:String).
GetProcessScenariosComplete	Similar to the previous function; returns all scenarios for a specified process id, but the result contains all details of each scenario, including all SLAs defined. Used for reporting purposes. The result is an indexed collection of named collections. Each named collection contains all data for a scenario plus the list of activity SLAs for the scenario. E.g. [{'scenarioID'='individual', 'scenarioName'='Individual', 'SLA'='10', 'threshold'='0.75', 'activities'= []}:String]:Any. The list of activities is another indexed collection of named collections. Keys for the scenarios are “scenarioID“, “scenarioName“, “SLA“, “threshold“. Keys for the activities are “procId“, “elemId“, “SLA“, “threshold“.
ExportSLADefinitions	Exports all SLA definitions to a specified text file. The default save location (unless otherwise specified) is {DATA_HOME}/work/BAM/SLAexport.txt. The content of the file is the java serialized version of all process and activity SLA objects; therefore it is not possible to edit the file manually. This feature is the same that is also available in the Appway Studio in the BAM Installation module. It is also available as script function for deployment automation purposes.
ImportSLADefinitions	

	Imports all SLA definitions from a specified text file that is the result produced by the ExportSLADefinitions function. This feature is the same that is also available in the Appway Studio in the BAM Installation module. It is also available as script function for deployment automation purposes.
DeleteBAMData	Deletes all BAM data related to processes completed or cancelled before the specified date (not included). Optionally, a process id can be specified so that only data related to that specific process are deleted. This feature is the same that is also available in the Appway Studio in the BAM Installation module. It is also available as script function for automation purposes.

Table 10: Utility Functions

3.3.2 Process Functions

This set includes functions that can be used inside processes in order to retrieve and use BAM information for a specific process instance.

Function	Defintion
GetProcessWarningTime	Returns the warning time for the specified process instance id as timestamp. This is a shortcut for reading from the process instance attribute "processWarningTime".
GetProcessExpirationTime	Returns the expiration time for the specified process instance id as timestamp. This is a shortcut for reading from the process instance attribute "processExpirationTime".
GetActivityWarningTime	Returns the warning time for the specified process token id as timestamp. This is a shortcut for reading from the process token attribute "activityWarningTime".
GetActivityExpirationTime	Returns the expiration time for the specified process token id as timestamp. This is a shortcut for reading from the process token attribute "activityExpirationTime".
SetProcessSLAScenario	

	Updates the current scenario id for a running process instance. This function updates the process warning and expiration time in the BAM DB and in the process attributes. The warning and expiration times of the running activities in the processes are not updated, meaning that the SLA for an activity in progress never changes. The scenario change will affect all new activities started after the scenario change. This also writes the current scenario id to a process instance attribute called "BAM_Scenario". Optionally, the name of this attribute can be changed in the extension configuration using the property bam.scenario.attribute.name.
GetProcessSLAScenario	Returns the current scenario id for the specified process instance id. This is a shortcut for reading from the process instance attribute "BAM_Scenario" (or what is contained in the property bam.scenario.attribute.name).

Table 11: Process Functions

3.3.3 Extraction Functions

The last set of functions includes the actual extraction functions that are used in the dashboards to read runtime process data. The functions use standard filtering structures as input parameters in order to standardize the way the results are filtered.

BAM provides two filtering structures: **BAMFilter** for process filtering and **BAMActivityFilter** for activity filtering. These are deployed as Appway dataclasses by the BAM extension and can be used as inputs when invoking the extraction functions from the scripting language.

The **BAMFilter** structure contains the following filtering criteria:

processId	process id to be considered (mandatory)
scenarioId	scenario id to be considered
processStatus	status of the processes to be considered ('Active'|'Completed'|'Cancelled') (mandatory)
startDateCreation	start date for creation time (Date)
endDateCreation	end date for creation time (Date)
startDateModification	start date for modification time (Date)

endDateModification	end date for modification time (Date)
attributes	named collection of process attributes names and values. For optimal query performances, at most 4 attributes should be specified.
dateFilterType	used for the out-of-the-box dashboards; can be left blank

Table 12: BAMFilter filtering criteria

The **BAMActivityFilter** structure contains the following filtering criteria:

taskId	id of the task
workflowId	id of the sub-workflow
userId	user id of the owner of the task
swimlaneName	name of the swimlane the activity is in
activityStatus	current status of the activity ('Active'|'Completed'|'Cancelled')
SLAStatus	current SLA status ('green'|'red'|'yellow')

Table 13: BAMActivityFilter filtering criteria

Here is the set of extraction functions:

Function	Description
CountInstancesByStatus	Returns the amount of instances in the specified status, given the BAMFilter. The result is an Integer.
CountInstancesBySLAStatus	Returns the amount of instances in the specified SLA status, given the status ('green'|'red'|'yellow') and the BAMFilter. The result is an Integer.
CountInstancesByScenario	Returns the amount of instances by scenario, given the BAMFilter. The result is an indexed collection of named collections. Each named collection contains a scenario, and the keys are: "scenarioId", "scenarioName", "count" e.g. [{'scenarioId'='indiv', 'scenarioName'='Individual', 'count'='10'}:String]:Any.
CountInstancesByMonth	

	Returns the amount of instances by month, given the BAMFilter. The result is a named collection with the months as keys and the count as values e.g. {'201302'=10, '201303'=12}:Integer. The months are in the format "YYYYMM". This function also allows the use of a multiplier for the instance number. The multiplier is specified as the name of a process instance attribute. For example, if more than one account can be opened inside one process, and the function should count accounts and not processes, a process instance attribute called numberOfAccounts could be used as multiplier for the number of processes.
ComputeAverageDurationByMonth	Returns the average instance duration by month in days, given the BAMFilter. It is also possible to specify the name of a process instance attribute to be used as multiplier (same as the function CountInstancesByMonth). The average will be weighted based on the multiplier.
GetInstancesOverview	Returns details about the set of processes instances that satisfy the given BAMFilter. It is also possible to specify a BAMActivityFilter; in that case the result will contain only instances with at least one sub-activity that satisfies the given BAMActivityFilter. The last input parameter (optional) is an indexed collection with the process instance attributes to be returned in the result. The return value is an indexed collection of named collections. Each named collection contains details about one instance. The keys are: "workflowId", "workflowInstanceId", "creationTime", "modificationTime", "processSLA", "processExpirationTime", "processWarningTime", "scenarioId", "scenarioName", "sla" ('green'|'red'|'yellow') plus the attribute names specified as input.
GetInstancesOverviewCount	Same as the function above, but only returns the number of instances that satisfy the given BAMFilter and, optionally, BAMActivityFilter
GetInstanceOverview	

	Returns details about a specific process instance, given the process instance id and (optionally) an indexed collection with the process instance attributes to be returned in the result. The result is a named collection with the keys: "workflowId", "workflowInstanceId", "status", "creationTime", "modificationTime", "processSLA", "processExpirationTime", "processWarningTime", "scenarioId", "scenarioName", "sla" ('green'|'red'|'yellow') plus the attribute names specified as input.
GetInstanceActivities	Returns the list of activities for a specific process instance, given the process instance id. The result is an indexed collection of named collections. Each named collection contains details about one activity. The keys are: "elementId", "workflowId", "elementName", "userId", "status", "creationTime", "modificationTime", "depth", "parentTokenId", "suspendedFor", "activitySLA", "activityExpirationTime", "activityWarningTime", "elementType", "elementInfo", "swimlaneName", "sla" ('green'|'red'|'yellow'), "workflowTokenId".
GetUserActivities	Returns the number of red, green, yellow and total activities assigned to each user which are included in process instances that satisfy the given BAMFilter. The result is an indexed collection of named collections. Each named collection contains details for one user. The keys are: "userId", "status", "total", "red", "green", "yellow" e.g. [{'userId'='nm', 'status'='Active', 'total'='20', 'red'='10', 'green'='5', 'yellow'='5'}:String]:Any.
GetActivityOverview	Returns activity details grouped by activity, swimlane, status and sla status. As for the other functions, the BAMFilter input defines the set of processes to be considered. The result includes the count for each group and the average duration. The result is an indexed collection of named collections. Each named collection contains details about one activity group. The keys are: "workflowId", "elementId", "subWorkflowId", "elementName", "swimlaneName", "status", "count", "averageDuration", "sla" ('green'|'red'|'yellow').
GetActivities	

	Returns full details about the set of activities that satisfy the given BAMFilter and BAMActivityFilter. It is possible to specify two additional parameters with the list of process instance attributes and process token attributes to be returned in the result. The result is an indexed collection of named collections. Each named collection contains details about one activity and the keys are: "workflowId", "workflowInstanceId", "processCreationTime", "processModificationTime", "processSLA", "processExpirationTime", "processWarningTime", "scenarioId", "scenarioName", "processSLAStatus" (‘green’|’red’|’yellow’), "elementId", "elementName", "userId", "status", "activityCreationTime", "activityModificationTime", "depth", "parentTokenId", "suspendedFor", "activitySLA", "activityExpirationTime", "activityWarningTime", "elementType", "elementInfo", "swimlaneName", "ActivitySLAStatus" ('green'|'red'|'yellow'), plus the instance attribute names and the token attribute names specified as input.
GetActivitiesCount	Same as the function above, but only returns the number of activities that satisfy the given BAMFilter and BAMActivityFilter.
BuildBAMFilterSQL	Returns the SQL "where" clauses built based on the given BAMFilter. The BAMFilter values found in the input object are translated to SQL conditions that can be appended to a custom query built in the scripting language. The generated SQL uses “wi” as alias for the BAM_WorkflowInstances table; therefore it is necessary to use that alias in the custom query to be built. The generated SQL contains blank spaces at the beginning and end of the string, and contains the "and" keyword at the beginning only. For example: ' and wi.workflowId = 'myProcess' '.

Table 14: Extraction Functions

4 Studio Modules

After having installed the BAM extension and reloaded the studio, a new top-level section appears in the Appway studio. The new section is called “Monitor” and contains 4 sub-sections:

- Service Level Agreements: link to the SLA tool that lets the user configure scenarios and SLAs for all processes and their activities.
- BAM Dashboard: displays the committed version of the BAM Dashboard process.
- BAM Test Data Generator: displays the test data generator interface enabling the generation of test data for the BAM dashboards. Used only for showcase and testing purposes.
- SLA Report: displays the out-of-the-box report of all scenarios and SLAs defined in the environment.

The following section of this document provides details for each.

As for all top level Appway Studio sections, it is possible to configure a security role so that only people in that role are able to access the section. For the Monitor area, the role to be configured is "BAMAdministrator".

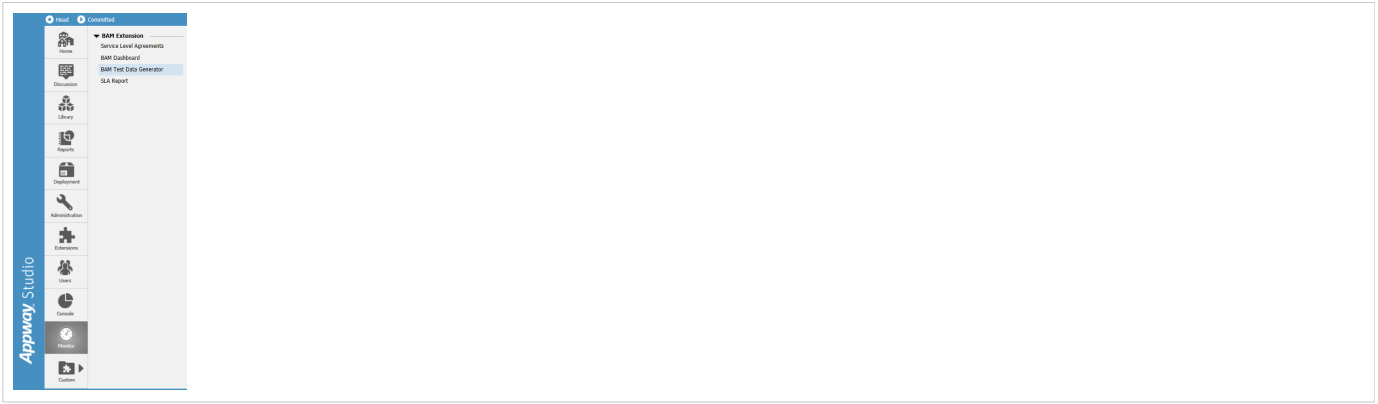


fig 14: BAM modules inside the Appway Studio

4.1 Service Level Agreements

This section allows the user to define scenarios and SLAs for Appway processes. As mentioned in the first section of this document, only activities and processes with an SLA defined (different from 0) will be saved in BAM and then made available in the dashboards. No process will be tracked in BAM by default. Therefore, setting the SLAs is a fundamental step in the setup of BAM.

4.1.1 Overview of Processes

The first screen of the SLA tool contains a list of all the processes that can be tracked in BAM. This list includes all processes defined in the library which have at least one swimlane. The reason for this limitation is that processes without swimlanes are temporary processes. They are not persisted beyond the user session and they are not even shown in the portal task list. Therefore, for those processes, it does not make sense to track any BAM data.

The list displays processes with two different icons depending on whether they have at least one SLA defined on them or none. See picture below.

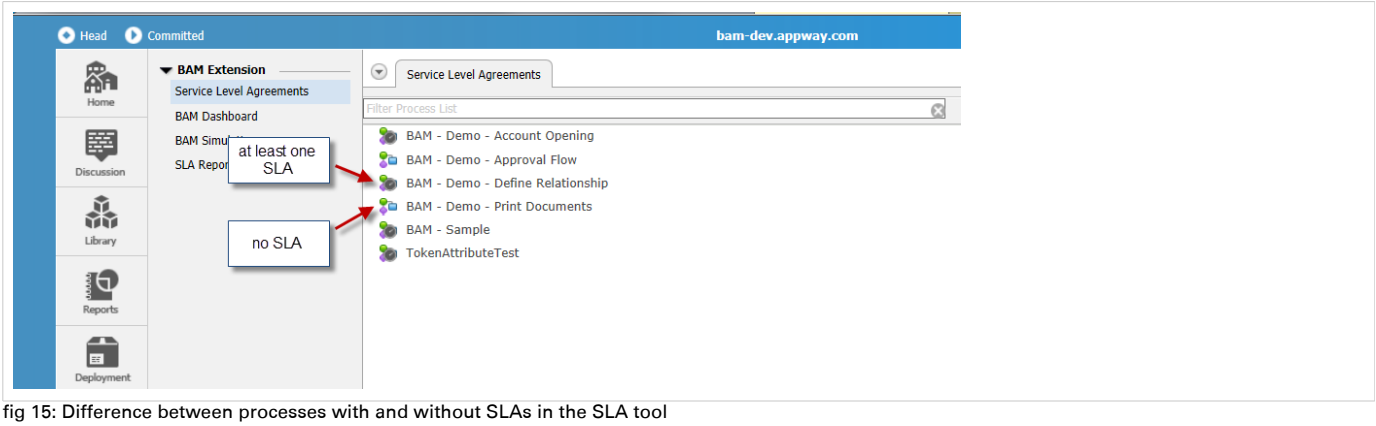


fig 15: Difference between processes with and without SLAs in the SLA tool

With a single click on a process in the list, the standard right bar is shown, and it is possible to access some details about the process, including id/name, file attachments, dependencies and change log. It is also possible to start the process in the workspace. This right-hand bar is the same that is also found in the library.

Upon right-clicking on a process in the list, an option appears to delete all scenarios and SLAs from the selected process.

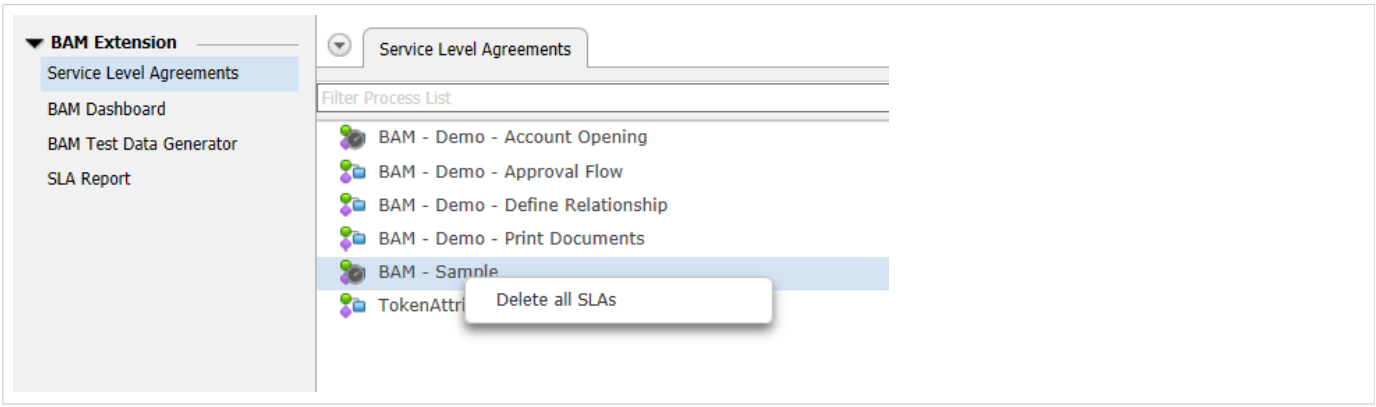


fig 16: Delete of all SLAs on a process from the SLA tool

4.1.2 SLA Editor

With a double click on a process, the actual SLA editor for the selected process opens in a new window. The editor is organized in a left-hand side bar used to configure scenarios and a right-hand side bar used to configure SLAs on single activities.

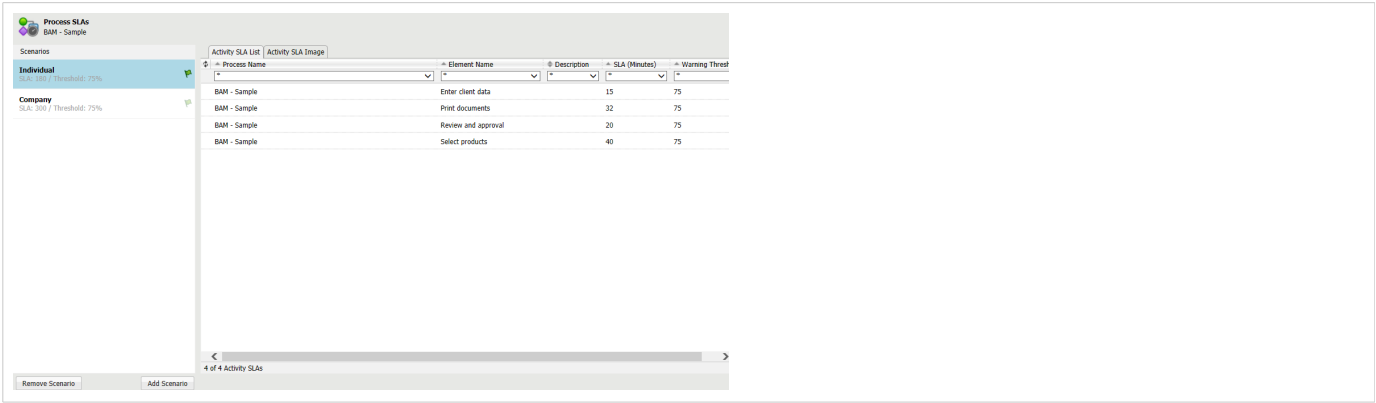


fig 17: Main page of the SLA tool showing scenario and SLA configuration

Scenarios

The scenario area lists all scenarios defined for the selected process, each including its SLA and threshold. The SLA value of a scenario indicates the overall expected duration of the process. By right-clicking on a scenario, the user can edit the scenario name, SLA (days, hours and minutes) and threshold. The warning time is automatically computed as SLA*threshold and displayed. The small green flag for each scenario indicates the default scenario. It is possible to switch from one default scenario to another by clicking on the flag of the desired default scenario. Each process instance where no scenario was set will be measured using the default scenario.

Two buttons at the bottom of the bar let the user remove an existing scenario and add a new one by specifying its id, name, SLA and threshold. The remove operation is a logical remove, meaning that the scenario is not deleted from the database, but just marked as deleted. This way, it will still be possible to browse and filter by deleted scenario in the dashboards. It will be not possible to assign a deleted scenario to a process, though.

For each process, a default scenario is automatically created by the BAM extension. The name of the scenario is "default" and the id is "1". If there is no need to specify different scenarios for a process, the default scenario can be used to set all SLAs. All processes will always be started under the default scenario if no other is available.

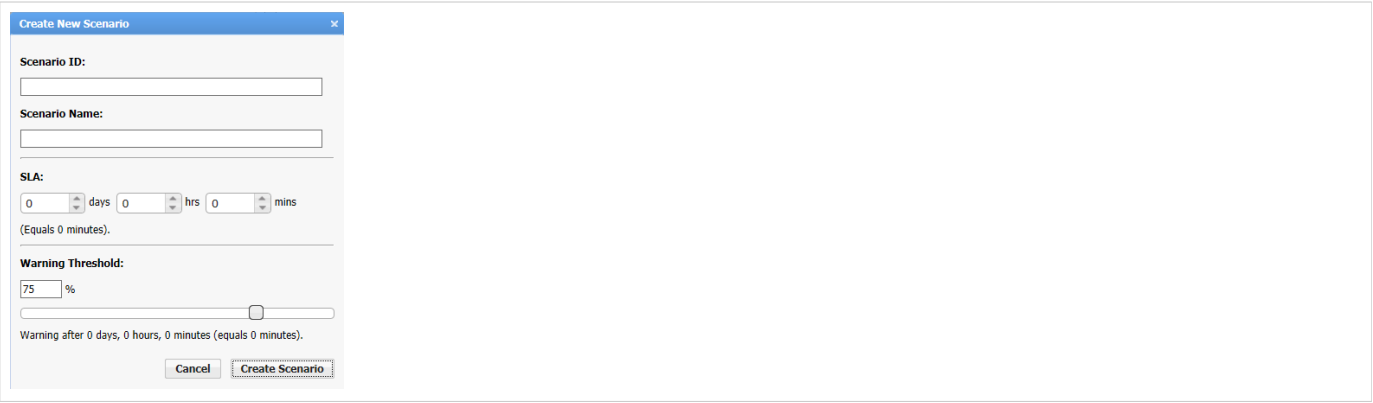


fig 18: Scenario creation pop-up

Activity SLA tabs

The activity SLA area includes two tabs: the "Activity SLA List" for quick entry of SLAs and thresholds from a sortable and filterable table; and the "Activity SLA Image", which provides a more graphical and intuitive interface.

The **Activity SLA List** presents all sub-activities and sub-processes of the previously selected root process in a flat list. The first column, "Process Name", indicates the name of the sub-process where the activity is located. It is possible to set the SLAs on a specific sub-process by selecting the sub-process name in the first column. All columns are sortable and filterable.

The SLA and threshold can be edited by double-clicking on the number to be edited. The SLA is in minutes and the threshold in percentage (0-100). An activity SLA equal to 0 is equivalent to no SLA set on the activity, meaning that the activity will not be saved to the BAM database.

The **Activity SLA Image** presents an image of the selected process with the SLAs and thresholds below each activity. With a right click on the image, it is possible to open the process in the process editor. Edit the SLA and threshold of an activity by simply clicking on it in the image. This will open an edit pop-up.

In case the process contains sub-processes, it is possible to drill down into one sub-process by right-clicking on it and selecting “Jump to this Sub-Process”. This will open the sub-process image, allowing the user to enter SLAs and thresholds for all activities in the sub-process. This can be repeated recursively for any nesting level. The desired sub-process can also be selected in the dropdown list at the top of the “Activity SLA Image” tab.

Finally, note that is possible to edit the SLA and threshold for all processes and activities at any time, but the change will only be applied to the new process instances and activities started after the change. In other words, a change to a SLA value will not be reflected in process instances that were already running when the change was applied.

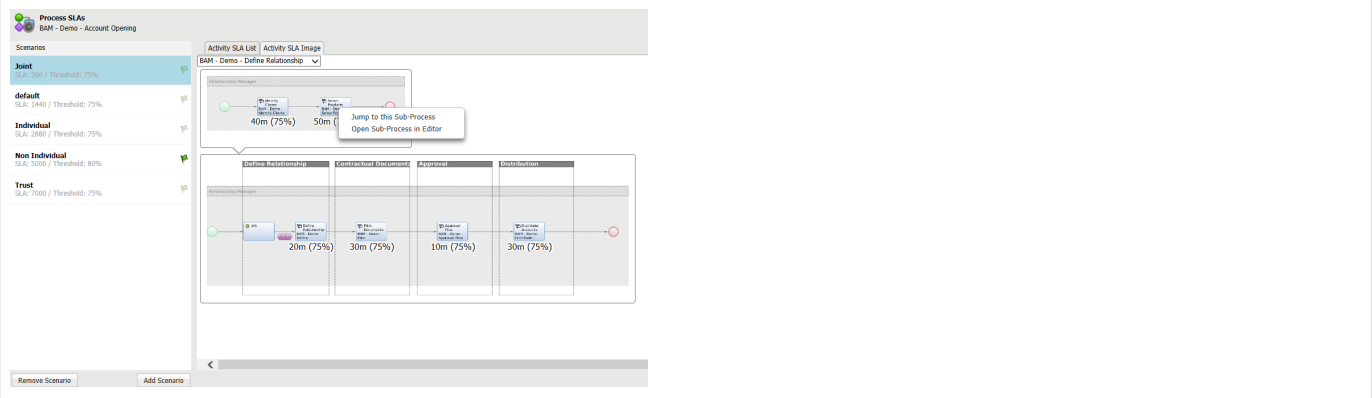


fig 19: Visual SLA editor with process drill-down feature

4.2 BAM Dashboard

This section gives quick access to the BAM Dashboards process. The content of this section is the same that would be generated when starting the committed version of the process in the workspace. By default, the process id is “BAMDashboards“, the id of the process in the out-of-the-box solution. This can be changed using the property bam.dashboard.process.

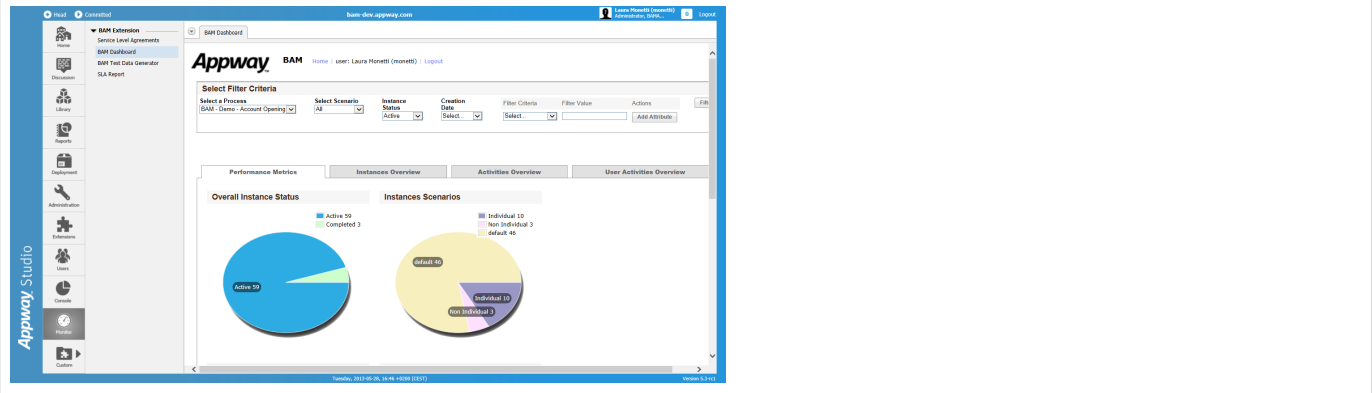


fig 20: BAM Dashboards displayed inside the Appway Studio

4.3 BAM Test Data Generator

This section displays the test data generation interface, which generates test data for the BAM Dashboards. This feature is used only for showcase and testing purposes. The section is by default disabled and can be enabled by setting the configuration property bam.testDataEnabled to true.

The process that is used for test data generation is called “BAM_Sample”, and is contained in the additional export package “BAMTestDataGenerator.awexport.” The package can be imported if test data generation capabilities are required. No other process is currently supported for test data generation.

The purpose of this module is to create test data for the BAM Dashboards by running an arbitrary number of test processes. As for all the other processes, in order to be able to save and display BAM data, it is necessary to set SLAs on one or more of the activities. Some sample SLAs for this process are defined in the export file “BAMTestDataGenerator.sla.” It can be imported using the “Import SLA” feature in the Administration section – BAM Installation module. After the import, the scenarios, SLA and threshold values can be changed if necessary.

The test data generation screen in the Appway Studio allows the selection of multiple parameters, including the number of test processes to be started (maximum 20 processes per run), the scenario to be applied, the users to be selected for the 2 swimlanes of the process and in which process step the test run should stop. It is also possible to specify the duration in minutes of all steps (“Think Time”). This duration will be correspondingly scaled down (by a factor of 600) so that the test run is always executed in a short amount of time.

The “Run” button at the bottom of the screen will start the test run and the generated data will be available in the dashboards under the “BAM - Sample” process. Once the showcase or testing is completed, the test data should be deleted with the “Clear Test Data” button.

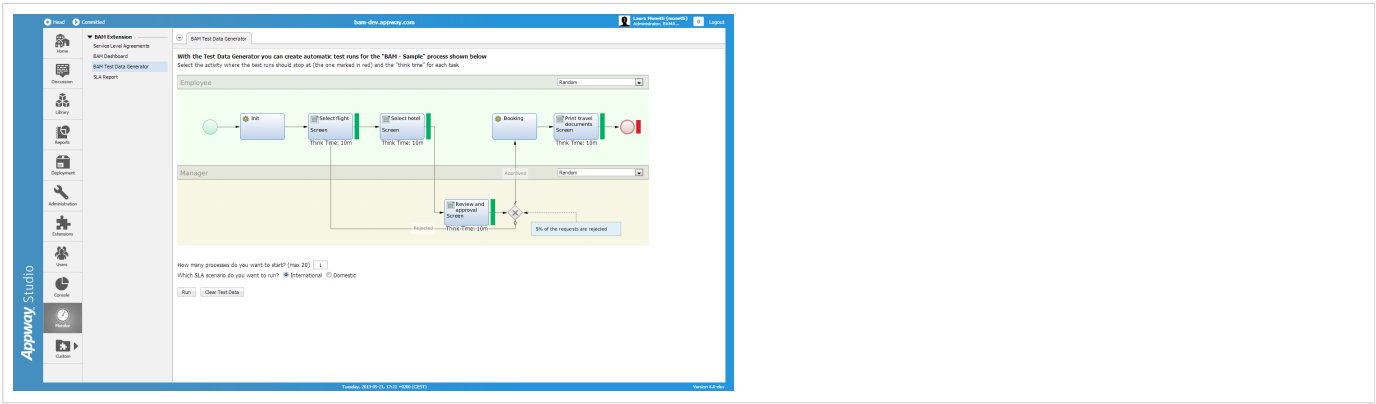


fig 21: BAM test data generator tool inside the Appway Studio

4.4 SLA Report

This section displays a PDF report of all the SLAs that have been defined in the current environment. The PDF Report is generated using the standard Appway Report functionality. Therefore, the report is defined as an Appway object and can be fully customized.

The configuration property “bam.slareport.report” of the BAM Extension allows the user to select a different report to be displayed in this section. The value of this property should be the id of a valid Report object in the Appway library.

In the out-of-the-box solution, the report is called “BAM_SLAReport” and is built with a dynamic document (PDF Output + PDF Output Table). It generates a PDF file with the list of processes on which at least one SLA is defined, each including the list of scenarios and activities SLAs.

The details for a specific process can be accessed by clicking on the content overview on the left. The details contain the image of the process being analyzed and one SLA table for each scenario, with the corresponding SLA and threshold. The SLA table reports the list of activities where an SLA is set, with the process name and element name for the activity, plus the SLA in minutes, the threshold as percentage and the calculated Warning SLA in minutes (Warning SLA = SLA * threshold%/100).

The PDF contains the information that can be accessed and configured in the SLA Management tool. It can be downloaded and saved to the local drive for reporting purposes.

The data is retrieved using the function GetProcessScenariosComplete.

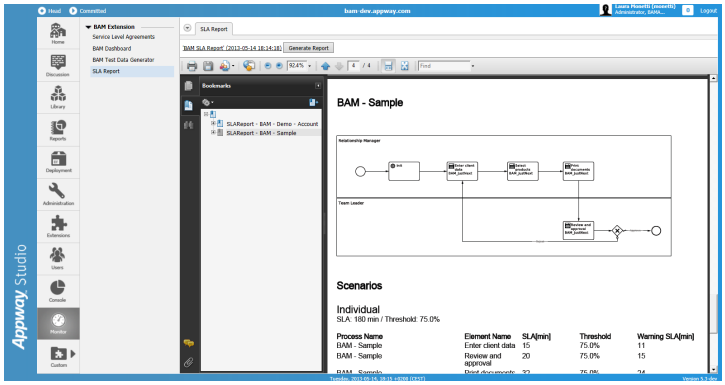


fig 22: BAM PDF SLA report inside the Appway Studio

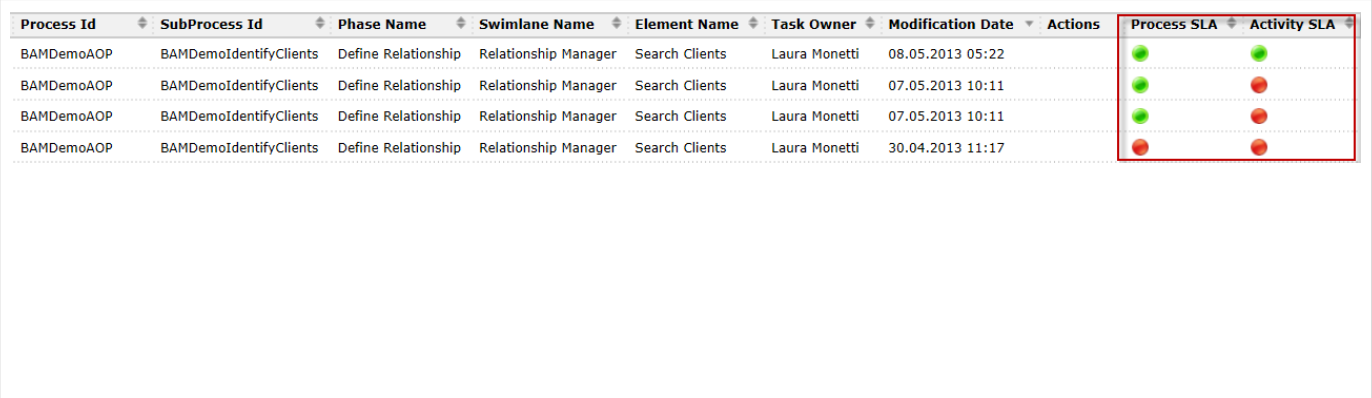
5 Additional Components

5.1 Portal Worklist SLA Columns

Appway applications' end users usually want to be able to check the status of the tasks assigned to them easily. They need to know if any activity is expired or close to expiration according to the SLAs.

For this purpose, make the SLA status of tasks visible on users' task list (or worklist) in the application's landing page, the portal.

For each task, it is recommended to display the SLA status of the activity as well as the SLA status of the overall process instance. Add two such columns in the task list with a marker flag as in the example below.



Process Id	SubProcess Id	Phase Name	Swimlane Name	Element Name	Task Owner	Modification Date	Actions	Process SLA	Activity SLA
BAMDemoAOP	BAMDemoIdentifyClients	Define Relationship	Relationship Manager	Search Clients	Laura Monetti	08.05.2013 05:22			
BAMDemoAOP	BAMDemoIdentifyClients	Define Relationship	Relationship Manager	Search Clients	Laura Monetti	07.05.2013 10:11			
BAMDemoAOP	BAMDemoIdentifyClients	Define Relationship	Relationship Manager	Search Clients	Laura Monetti	07.05.2013 10:11			
BAMDemoAOP	BAMDemoIdentifyClients	Define Relationship	Relationship Manager	Search Clients	Laura Monetti	30.04.2013 11:17			

fig 23: BAM "traffic-light" icons displayed using the Portal Worklist SLA columns

Appway provides a standard task list component in the PortalWorklist extension. This component includes two out-of-the-box columns for the process and activity SLA status.

Installing the BAM and the PortalWorklist (version 2.2.8 or higher) extensions are prerequisites. With both installed, it is possible to select "ProcessSLA" and "Activity SLA" as column types when creating a Worklist Column inside a Worklist component in a screen.

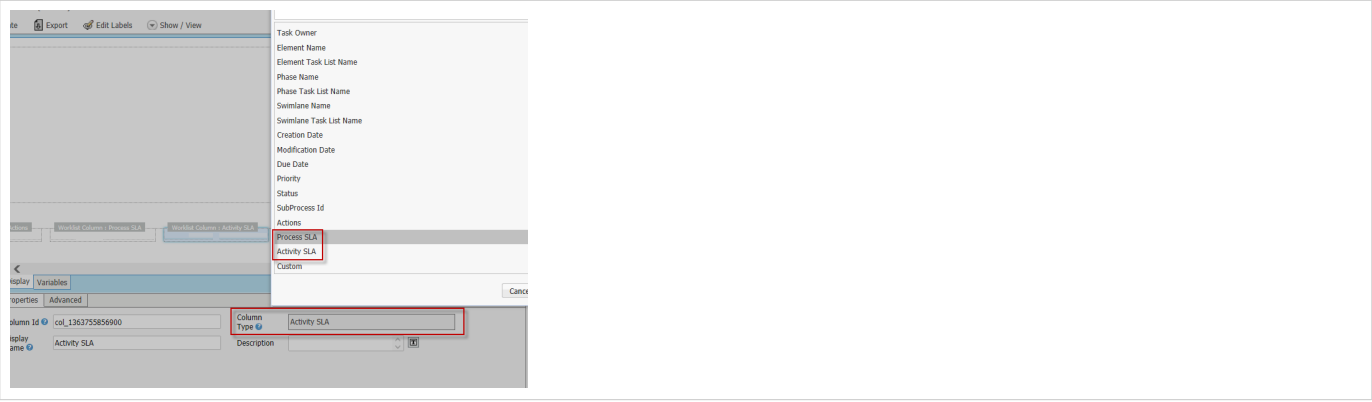


fig 24: BAM column selection inside the Portal Worklist Column component

The out-of-the-box behavior of the Process SLA column is that the PortalWorklist extension will read the process warning and expiration time from the task. It will then generate the "traffic light" icon based on the following rules:

Icon Color	Rule
grey	If the process warning or expiration times are null or equal to 0, the extension will assume the SLA was not defined on the process and it will return a grey icon. The tooltip for the icon is “SLA not set”.
red	If the current timestamp is higher than the process expiration time, the extension will generate a red icon. The tooltip for the icon is “Expired for: d:h:m:s” with the actual amount of time the process has been expired for (timestamp - expiration time).
green	If the current timestamp is lower than the process warning time, the extension will generate a green icon. The tooltip for the icon is “Remaning: d:h:m:s” with the actual amount of time remaining before expiration (expiration time - timestamp).
yellow	In other cases, the extension will generate a yellow icon, meaning that the process is not expired yet, but the threshold has been exceeded, and the process is therefore in “warning” status. As in the “green” case, the tooltip is “Remaning: d:h:m:s” with the actual amount of time remaining before expiration (expiration time - timestamp).

The same behavior is implemented for the Activity SLA column, the only difference being that the activity warning and expiration time are used for evaluating the status of the task. In case an SLA is not set on the specific activity, Appway will travel up the sub-process hierarchy in order to identify the lowest defined SLA in the container sub-processes.

6 Out-of-the-Box Dashboards

6.1 Prerequisites

Some prerequisites must be satisfied for the out-of-the-box dashboards to function correctly. These consist in installing three extensions:

- ChartingExtension, version 2.0.46 or higher
- WorkflowImage extension, version 1.0.3 or higher
- Component extension (tested with version 2.0.75)
- POI extension (tested with version 2.1.5), configured as follows:

```
template.activities.resource.id = BAM_ActivitiesReport
template.activities.start.x = A
template.activities.start.y = 5
template.instances.resource.id = BAM_InstancesReport
template.instances.start.x = A
template.instances.start.y = 5
```

6.2 Installation

The out-of-the-box dashboards are shipped inside the BAM extension as an .awexport content package. This package can be imported from the Administration – "BAM Installation" module into the Studio by clicking on the **"Install Dashboards"** button. It will open the standard import interface in order to give the possibility to check the result of the import.

This button is used for the first installation of the out-of-the-box dashboards: it imports all content including the dashboard templates and the default settings. All imported objects should be committed, either automatically during the import or manually immediately afterwards.

After the first installation, clicking on the "Install Dashboards" button again will result in resetting the status of the dashboards to the initial one, overwriting all customizations. The conflicts with local changes will be marked in the import dialogue.

6.3 Upgrade

When a new compatible version of the BAM extension is installed, it is possible to update the dashboards to take advantage of the new features that have been implemented. Use the **"Update Dashboards"** button in the Administration section – "BAM Installation" page. It imports the dashboard templates only and opens up the standard import interface in order to check the result of the import and resolve any conflict.

In case no customization was performed on the out-of-the-box dashboard objects, the import will merely upgrade the dashboards. In case a customization was carried out following the customization guidelines outlined in next chapter ("Dashboards customization"), the upgrade will install the new features while still preserving the customizations. In these two cases, Appway will not highlight any conflict during the import procedure.

In case of customizations not compliant with the guidelines, there is no guarantee that it will be possible to keep the customizations after the upgrade. Conflicts might come up during the import procedure which must be analyzed and resolved case by case in order to retain the desired functionalities. If the conflicts are ignored, there is no guarantee that the customizations will be retained and the dashboards will be functional after the import.

6.4 Filters

The out-of-the-box dashboard screens are organized in two sections: a top section for specifying a set of filtering criteria, and a central section showing the BAM data filtered according to the selected values.

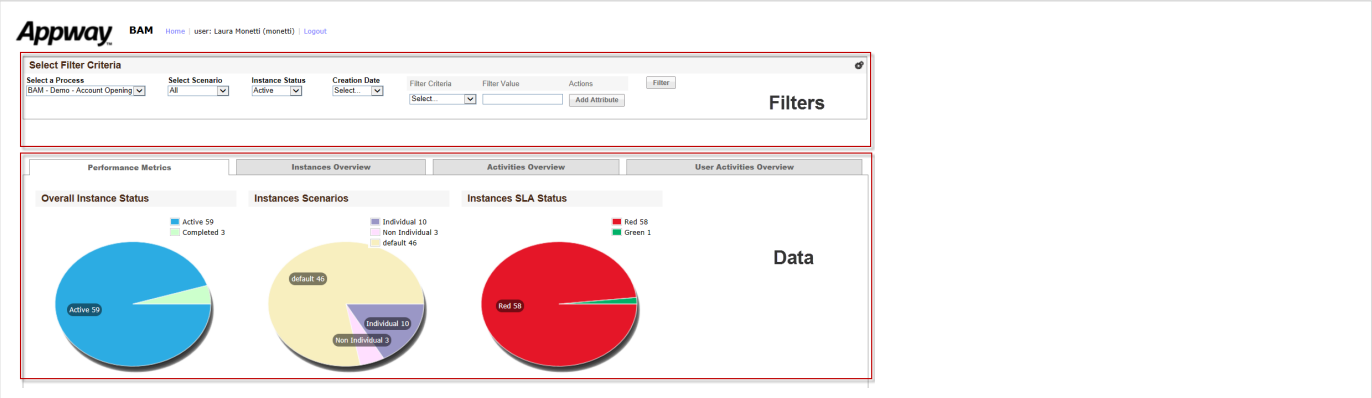


fig 25: BAM Dashboards: filters and data areas

The filtering area contains a set of parameters that correspond to the BAMFilter object described in the “Functions” section of this document. The filtering parameters are:

Parameter	Description
Process	Name of the process to be monitored The selection of a process is mandatory: it is not possible to browse data coming from different processes at the same time. The selection list includes all processes with at least one SLA defined on them.
Scenario	Name of the SLA scenario It is possible to select “All” to browse data executed under all scenarios. The list includes all scenarios that have been defined in the SLA tool for the selected process.
Instance status	Status to take into account ('Active'|'Completed'|'Cancelled') The selection of a status is mandatory: it is not possible to browse data coming from processes with different statuses at the same time.
Creation date	

	Date when the process was started (only if the "Active" status is selected) A set of predefined ranges is available (this week, last week, this month etc.) but it is also possible to specify a custom range by selecting a start and end date.
End date	Date when the process was completed or cancelled (only if the "Completed" or "Cancelled" status is selected) A set of predefined ranges is available (this week, last week, this month etc.) but it is also possible to specify a custom range by selecting a start and end date.
Filtering attributes	List of business attribute names/values The list of names available is configured in a dedicated catalog for each process, as described in the section "Business Attributes Configuration". The value is specified using a text field if the attribute contains a string or number, or using a time range if the attribute contains a date or time. If the attribute is bound to a set of values contained in a catalog, a dropdown is used to specify the attribute value. Up to four filtering attributes can be specified. The attribute names and values are compared to the process instance attributes set inside the process instances. Only the instances which contain attribute values matching the filters are returned in the results.

Table 16: Dashboard Filtering Parameters

The data in the lower part of the dashboards is filtered only when clicking the "Filter" button, and not upon each change of filtering criteria. This is to avoid unnecessary queries slowing down the execution.

In the top right corner of the filter section, a preferences icon lets users save their preferred filters. It provides the option to save the currently configured filter with a name, or to restore a previously created filter by selecting from a list of existing filter names. This way, if a user always needs the same set of filtering criteria, he does not need to reselect all the options every time he accesses BAM.

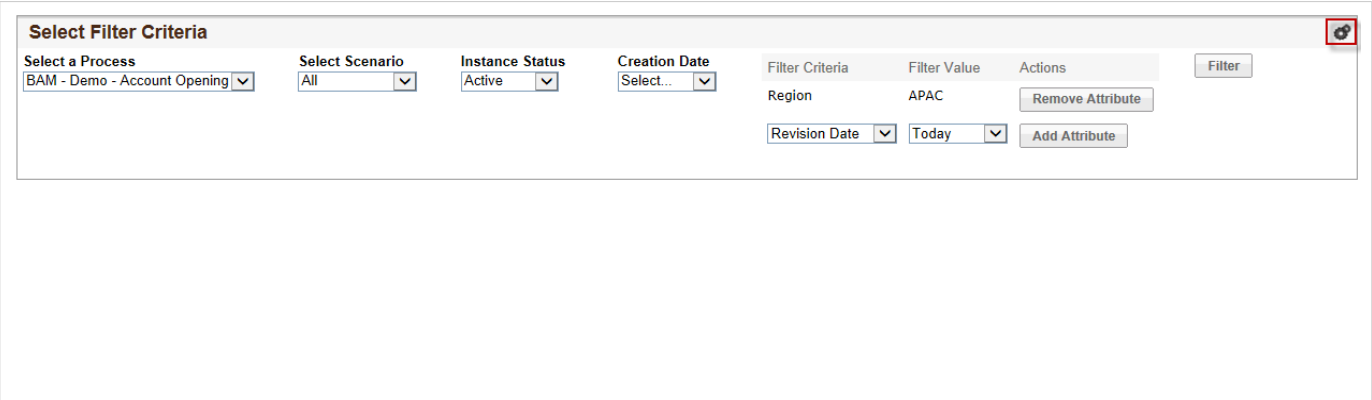


fig 26: BAM Dashboards: filtering panel

6.5 Dashboard Tabs

The central part of the dashboards presents BAM data that satisfy the selected filtering criteria. It is organized in four tabs: "Performance Metrics", "Instances Overview", "Activities Overview", and "User Activities Overview".

6.5.1 Performance Metrics

Performance Metrics presents an overview of the process instances for the selected process. In particular, it shows five monitoring charts:

Chart	Description
Overall Instance Status	This pie chart displays the amount of instances according to their status ('Active'|'Completed'|'Cancelled'). The chart is clickable: when clicking on a status section, the filter on top is automatically updated to the selected status and all the data in the dashboard are filtered accordingly. The data is retrieved using the function CountInstancesByStatus.
Instance Scenarios	This pie chart displays the amount of instances corresponding to each scenario according to the selected status. The chart is clickable: when clicking on a scenario section, the filter on top is automatically updated to the selected scenario and all the data in the dashboard are filtered accordingly. The data is retrieved using the function CountInstancesByScenario.
Instances SLA Status	This pie chart displays the amount of instances currently in each SLA status ('Red'|'Green'|'Yellow') according to the selected process status. The data is retrieved using the function CountInstancesBySLAStatus.
Number of Instances by Month	

	This bar chart displays the amount of instances in the selected status counted by month. If the selected status filter is “Active”, the chart will display the instances started in the given month, otherwise the ones that were completed/cancelled in the given month. The data is retrieved using the function CountInstancesByMonth.
Average Instance Duration by Month	This line chart displays the average monthly duration in days of process instances that are in the selected status. If the instances are still active, it computes the duration until the current time, otherwise the total duration. The data is retrieved using the function ComputeAverageDurationByMonth.

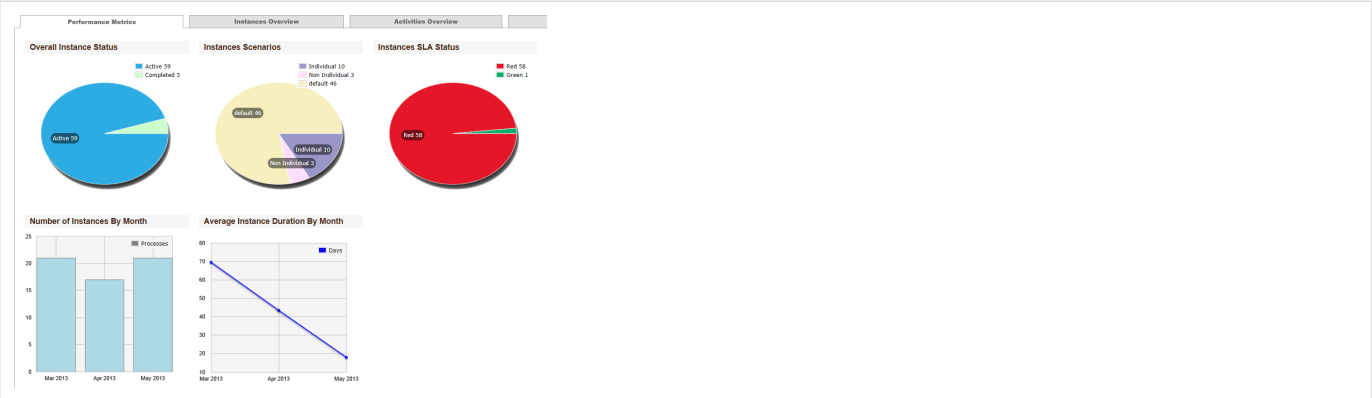


fig 27: BAM Dashboards: Performance Metrics tab

6.5.2 Instances Overview

Instances Overview presents a list of process instances filtered according to the selected criteria. Each row in the table represents one instance and shows basic information plus custom business attributes. For each instance, the table displays the process id (RequestId), scenario name, instance start date and time, remaining time before expiration and SLA status. Additionally, an arbitrary number of custom (process-specific) business attributes can be shown in corresponding columns.

The list of attributes to be included is configured in a dedicated catalog for each process, as described in the section “Business Attributes Configuration”. The table provides sorting, configurable paging and a global search field which searches through all the columns of all available instances. The data is retrieved using the function GetInstancesOverview.

The section has a limit on the maximum amount of instances that can be displayed in order to avoid performance issues. The default value is 500, but this can be edited by setting the Appway configuration property bam.instances.max.results. The number of instances is computed using the function GetInstancesOverviewCount.

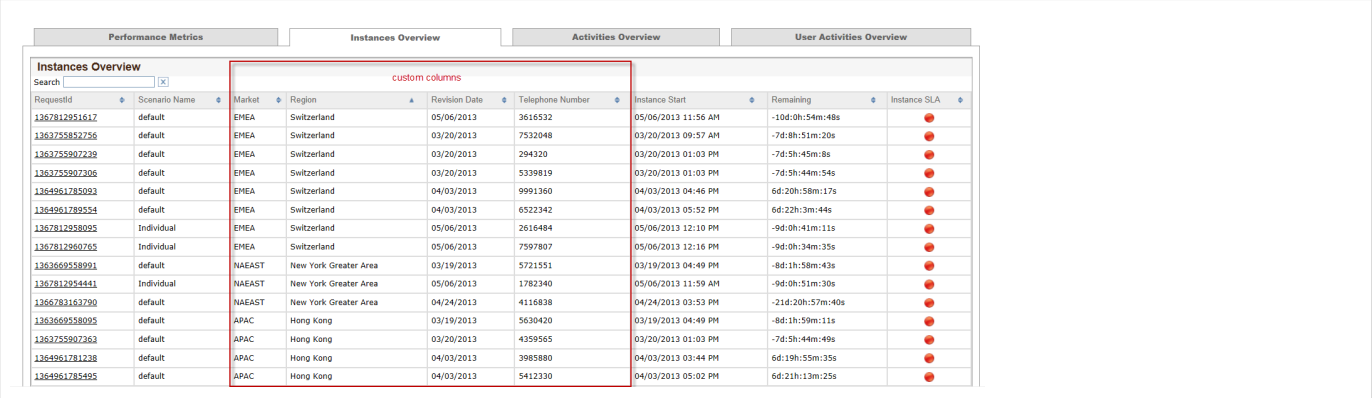


fig 28: BAM Dashboards: Instances Overview tab

Click on a row's process id to drill down into details about the specific process instance in the "Activity Details" screen. The details on the left include a list of the instance's standard and custom properties. Standard properties are process, scenario name, status, creation date and last modification date. In addition, the section shows a list of custom business attributes. The list of attributes to be included is configured for each process in a dedicated catalog, as described in the section "Business Attributes Configuration".

The data of the left section is retrieved using the function GetInstanceOverview. On the right, a table reports a list of all activities that have been performed or are being performed for the selected process. The table includes for each activity the step, role, status, owner (user or team), remaining time, SLA status, start time and (if completed) end time. The activities are indented basing on their nesting inside subprocesses of the main process. The data of the right section is retrieved using the function GetInstanceActivities.

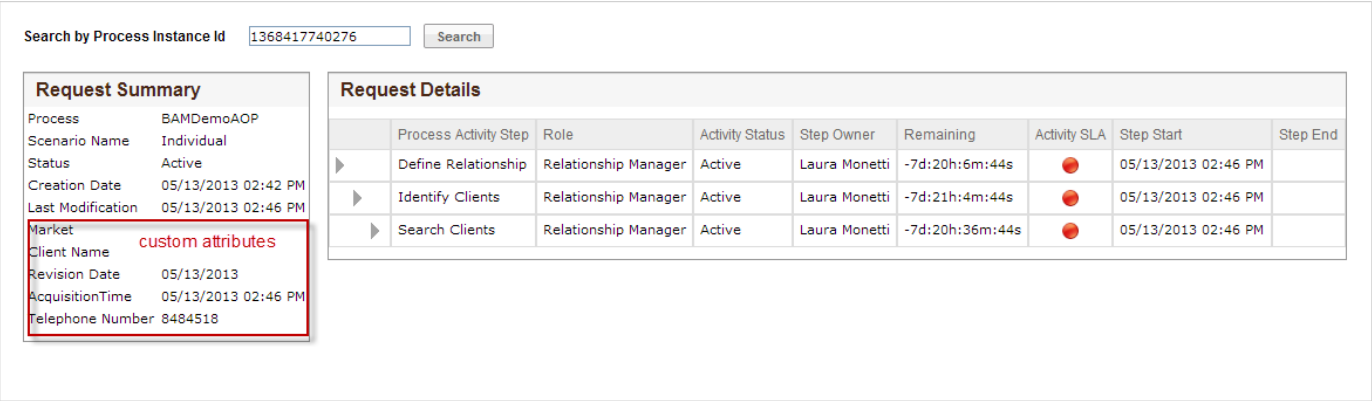


fig 29: BAM Dashboards: Activity Details screen

6.5.3 Activities Overview

Activities Overview presents a list of grouped activities. The activities are grouped by four criteria: task, role, status and SLA status. This tab of the BAM presents these groups in a table together with the average duration inside the group and the number of activities in the group. As for the previous tab, the table provides sorting, configurable paging and a global search field which searches through all the columns of all the available activities. The data is retrieved using the function GetActivityOverview.

It is possible to drill down into the details of a specific group by clicking on the count of activities. This will open details about all the relevant process instances that include the activities in the selected group. The view on the instances is the same as in the previous tab "Instances Overview", but only containing filtered instances. Also similarly to the previous tab, it is possible to drill down further into single instances. The data is retrieved using the function GetInstancesOverview.

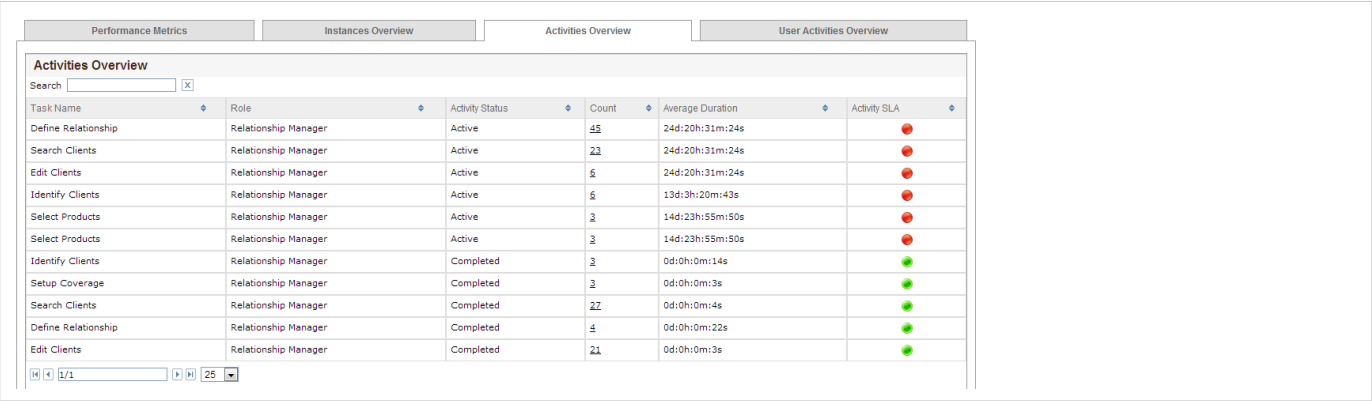


fig 30: BAM Dashboards: Activities Overview tab

6.5.4 User Activities Overview

User Activities Overview presents a view on activities focused on users. The tab includes activities grouped by user and status, including total number and number of red, green and yellow activities. As for the previous tabs, the information is presented in a table which provides sorting, configurable paging and a global search field which searches through all the columns of all the available activities. The data is retrieved using the function GetUserActivities.

It is possible to drill down into the details of a specific group by clicking on the count of activities. For example, browse the details of all active activities assigned to a specific user and for which the status is "red" (expired). This will open up details about all the activities in the selected group, displaying activity information together with instance level data. The columns of this table include process instance id, instance SLA status, custom instance level attributes, activity step, role, status, owner (user or team), remaining time, activity SLA status, start time and (if completed) end time. The data is retrieved using the function GetActivities.

This tab has a limit on the maximum amount of activities that can be displayed in order to avoid performance issues. The default value is 1000, but this can be edited by setting the Appway configuration property bam.activities.max.results. The number of activities is computed using the function GetActivitiesCount.

Similarly to the previous tabs, it is possible to drill down further into single instances by clicking on the process instance id.

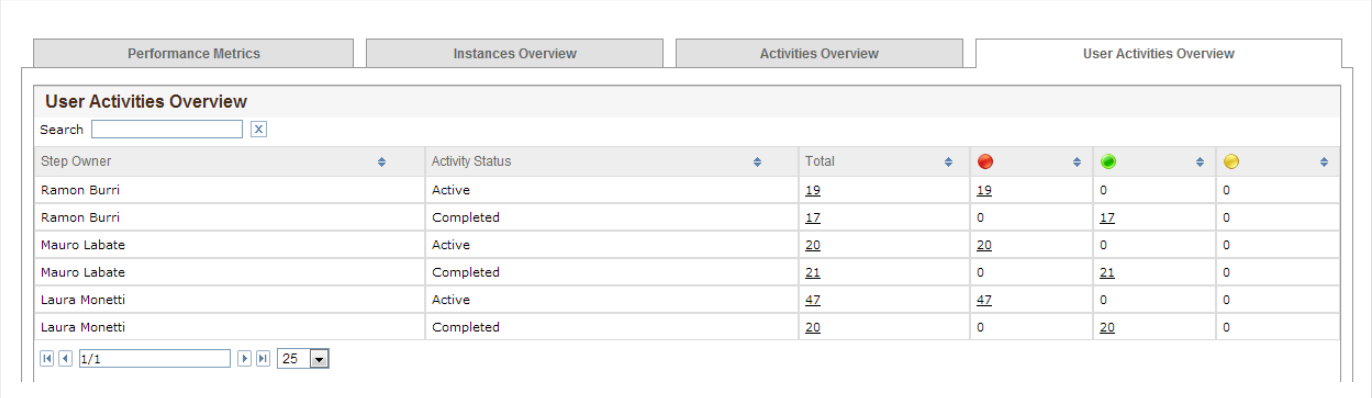


fig 31: BAM Dashboards: User Activities Overview tab

6.6 Business Attributes Configuration

As seen in the previous section, the dashboards can contain custom business indicators defined for a specific process. The set of instance attributes to be used in the dashboards for this purpose must be defined in the catalog "BAM_Attributes", which is created while importing the out-of-the-box dashboards.

The catalog contains one row per attribute and includes the following columns:

Column Name	Description
id	id of the attribute as it is saved in the process instance
description	user-friendly description to be displayed in the dashboard screens
filter	defines whether the attribute can be used for filtering in the attribute filtering section of the filters bar ('true'|'false') If true, the attribute will appear in the attribute's dropdown selector.
display	defines where the attribute is displayed ('overview'|'details'|'full') If "overview", the attribute is displayed in the instances overview tab only. If "details", the attribute is displayed in the instance details only. If "full", the attribute is displayed in both cases.
processes	

	defines the list of processes the attribute is relevant for Contains a single process id, or a pipe-separated list of ids if the attribute is relevant for multiple processes, or "*" if the attribute is relevant for all processes. The attribute dropdown in the filtering area and the data tables will only contain attributes that are valid for the selected process.
type	defines the type of attribute and is only mandatory for Date and Time attributes ('Date'|'Time'|empty) If the attribute is not of one of these two types, the column can be left blank.
catalog	the id of the catalog to be used as a source for the list of possible values of the attribute. If a catalog id is specified, the filtering area will present a dropdown instead of a text field for the attribute value selection. Also in all places where the attribute value must be displayed (tables, reports, filters), the value will be decoded using the specified catalog: the actual value will be used as 'id' in the catalog and the 'description' column will be shown instead.

Table 18: Configuration of the BAM_Attributes catalog

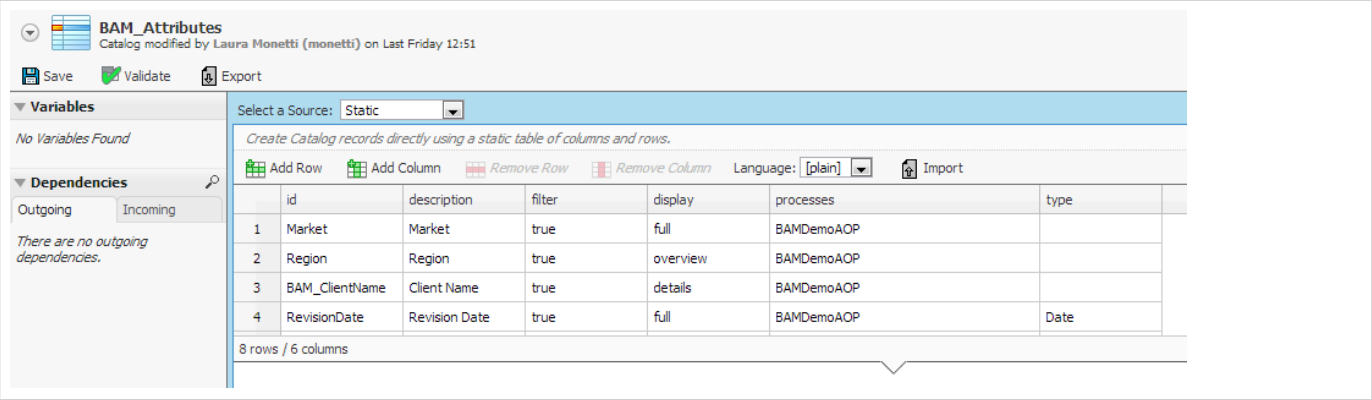


fig 32: BAM_Attributes catalog to configure custom business indicators

6.7 Trackpoints

Once the attributes have been defined in the BAM_Attributes catalog as described in the previous section, it is possible to set a value for each inside the relevant processes. The BAM extension provides a visual way to set BAM-relevant process instance attributes as opposed to using Appway scripting language (standard approach). This feature is based on BAM Trackpoints, a dedicated component available in the process editor.

A BAM trackpoint allows the user to set a BAM-relevant process instance attribute to a static value or an expression by placing the trackpoint on an element or connection inside the process. The attribute will be set to the selected value in the process instance when a token enters the element or connection.

The trackpoint properties include a name, a description, the attribute name and the attribute value. Select the attribute name from a dropdown that includes all attributes that are listed in the BAM_Attributes catalog for the current process (see “processes” column). The attribute value can be a static value or the result of the evaluation of an expression.

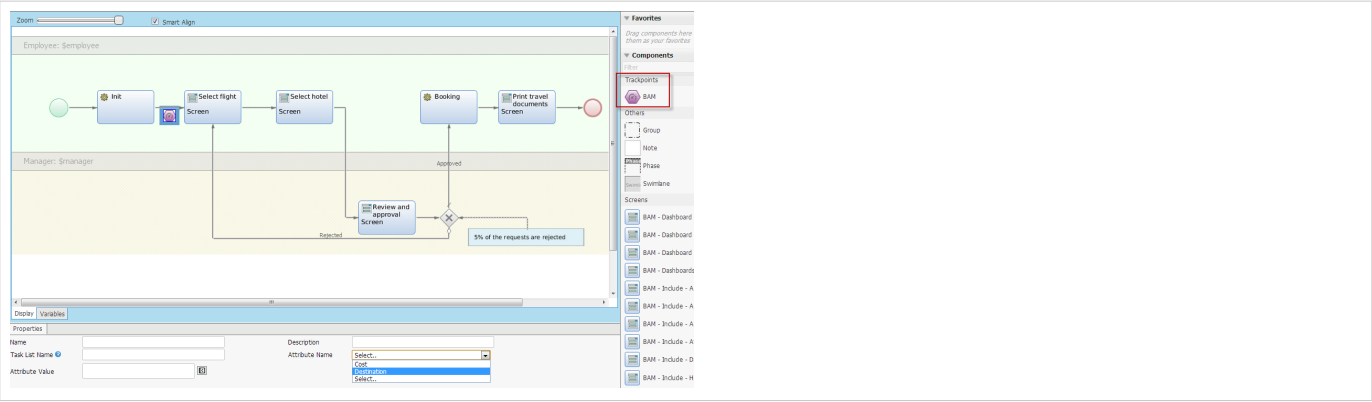


fig 33: Example of BAM trackpoint configuration inside the process modeler

6.8 Reports

The out-of-the-box dashboards include a spreadsheet-based reporting feature. It is possible to export full or filtered sets of instances and activities into MS Excel.

6.8.1 Instance Report

The instances report contains data about a set of process instances. The report is generated with the POI extension using the template “BAM_InstancesReport”. Download the report from the Instances Overview tab by clicking on the button “Download Instances Report” at the bottom of the page. It will contain data about all instances which satisfy the filters specified in the top filtering bar (process, status, etc.).

The same can be achieved in the Instance Overview page accessed by clicking on a group count in the Activities Overview. In this case, the report will contain data about the subset of selected instances, i.e. those that contain an activity belonging to the selected group.

The columns of the spreadsheet generated include basic instance information (process instance id, process id, status, scenario name, instance start, remaining time, instance SLA) plus all custom business attributes defined for the selected process, independently of the value that was set for the attribute in the “display” column of the BAM_Attributes catalog. The data is retrieved using the function GetInstancesOverview.

6.8.2 Activities Report

The activities report contains data about a set of activities. The report is generated with the POI extension using the template “BAM_ActivitiesReport”. Download it from the Activities Overview or User Activities Overview tab by clicking on the button “Download Activities Report” at the bottom of the page. The report will contain data about all activities that satisfy the filters specified in the top filtering bar (process, status etc).

The same can be achieved in the Activities Overview page accessed by clicking on a group count in the User Activities Overview. In this case, the report will contain data about the subset of selected activities, i.e. those that belong to the selected group.

The columns of the spreadsheet generated include basic instance information (process instance id, process id, instance status, scenario name, instance start, instance remaining time, instance SLA) plus all custom business attributes defined for the selected process, independently of the value that was set for the attribute in the “display” column of the BAM_Attributes catalog, plus activity information (activity, role, activity status, owner, activity start and end time, activity remaining time, activity SLA). The data is retrieved using the function GetActivities.

7 Dashboard Customization

7.1 Screen Structure and Templates

In order to be able to customize the BAM Dashboards, it is important to understand how they are modeled. This section describes the architecture of the dashboards.

The out-of-the-box dashboards are implemented with a process called “BAM - Dashboards”. It always displays the “BAM - Dashboards” screen, unless there is no SLA configuration at all in the system in which case it shows an error screen.

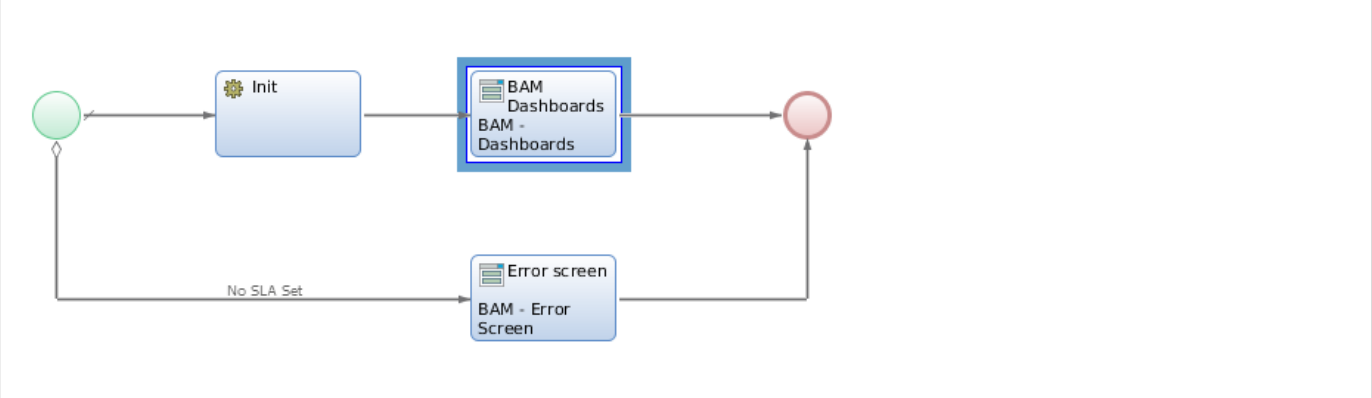


fig 34: “BAM – Dashboards” process

The “BAM - Dashboards” screen contains only a single screen template component (“Template – BAM - Dashboards”), which is the implementation of the out-of-the-box features. This template provides multiple areas where developers can include additional custom features, like additional tabs, charts or a custom header.

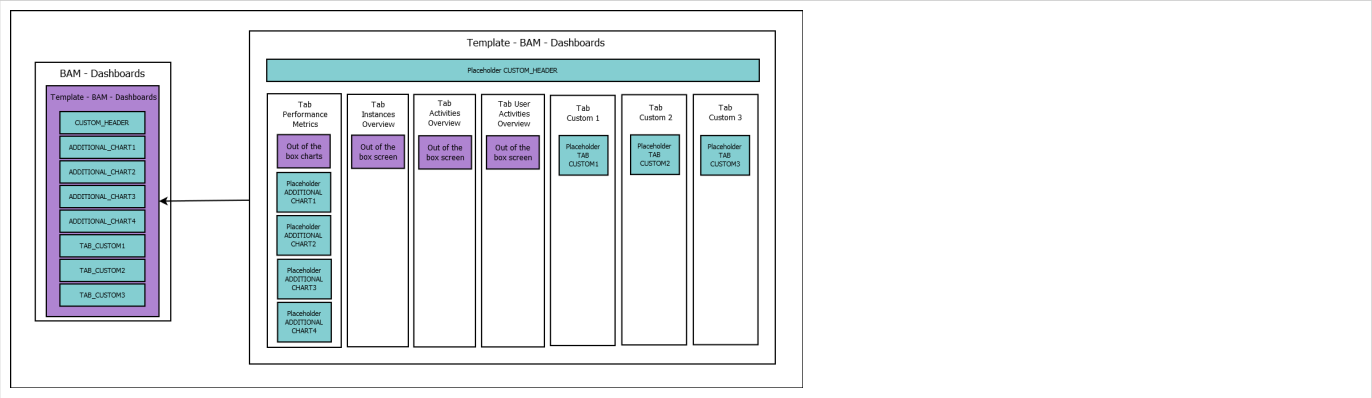


fig 35: Customization mechanism: high-level structure of the “BAM – Dashboards” and “Template – BAM – Dashboards” screens

The “BAM - Dashboards” screen is the screen to modify in order to customize the dashboard behavior.



fig 36: "BAM – Dashboards" screen implementation in the screen editor

"Template – BAM – Dashboards" in turn relies on the main "Template – BAM" screen which defines the frame of the application. The frame is made of three sections: header, filters and content. The header is the top bar of all the dashboard screens. By default, it contains the Appway logo, the home button and the logout button; this can be fully customized as described later in this chapter.

The filters are placed right below the header, providing filtering capabilities for the dashboards. The only aspect that can be customized in the filters is the list of custom business attributes as described in the previous chapter.

The content area of the "Template – BAM" screen includes the main content of the page, which changes based on which screen is being displayed.

The "Template – BAM" screen also includes a "BAM - Include - Header Includes" screen that imports the required Javascript libraries and CSS resources. The CSS resources can be customized as described later in this chapter.

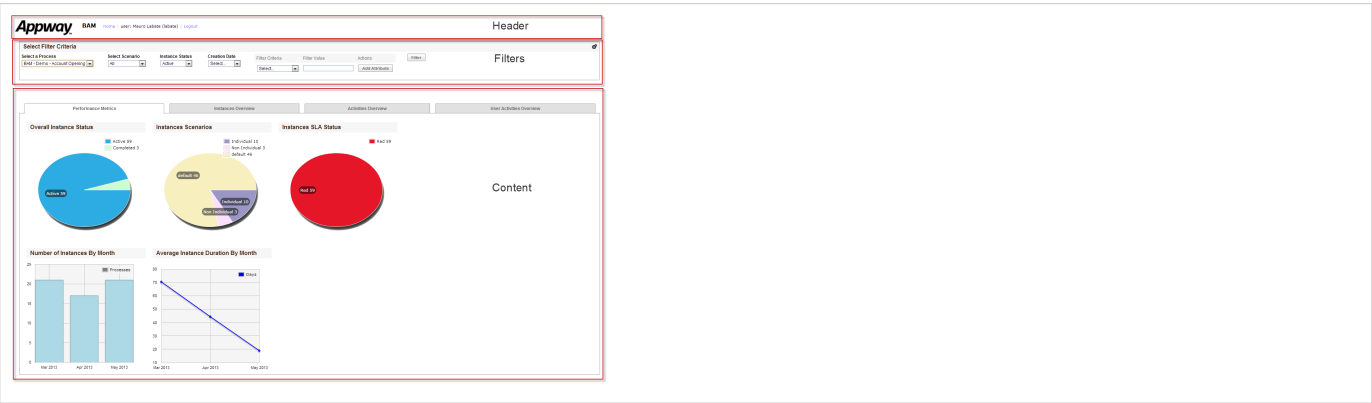


fig 37: BAM Dashboards sections including header, filters and main content areas

"Template – BAM – Dashboards" includes seven tabs containing the different sections of the dashboards. In the out-of-the-box solution, four of the seven tabs are already filled with content, as described in the previous chapter of this document. Three of them are available for custom content and can be enabled following the guidelines in the next section of this chapter.

7.2 Supported Customization Types

The out-of-the-box dashboards are a set of Appway business objects, so it is possible to fully customize them according to the specific project needs. However, if these business objects are changed without following any guidelines, upgrades of the BAM extension might result in conflicts and incompatibilities. For this reason, a set of commonly required customizations has been identified. They are supported so as to avoid conflicts when installing a new compatible version of the BAM.

The supported customizations include:

- Change of the top header of the BAM Dashboards
- Change of the CSS used in the BAM Dashboards
- Creation of a new chart in the "Performance Metrics" dashboard (up to 4 new charts)
- Removal of one or more of the out-of-the-box charts in the "Performance Metrics" dashboard
- Creation of a new dashboard tab (up to 3 new tabs)
- Removal of one or more of the out-of-the-box tabs
- Definition of a custom actions column in the "Instances Overview" tab
- Definition of a custom actions column in the "Activity Details" screen

The dashboards are designed in a way that allows implementing all these customization types by modifying a small subset of the BAM business objects:

- Screens:

  - BAM - Dashboards

  - BAM - Include - Activity Actions

  - BAM - Include - Instance Actions
- Catalogs:

  - BAM_DefaultFeatures

These business objects are not changed when updating the BAM Extension to a new version.

Any modification to the out-of-the-box dashboards which requires changing other screens leads to conflicts when updating the BAM Extension to a new version: for example, changing the "BAM - Include – DashboardFilters" screen to implement a custom filtering condition. In such a case, the options are:

1. Never install any update of the out-of-the-box BAM Dashboards after customization
2. Overwrite all the local customization with the new version when installing a new update of the out-of-the-box BAM Dashboards (therefore losing the customization work done previously)
3. Carefully solve the conflicts between the two versions (local changes and new out-of-the-box version) by merging all the business objects manually. This activity must be performed in close collaboration with Appway R&D department and there is no guarantee that a new BAM version can still support the local customizations.

Configuration catalog

The “BAM_DefaultFeatures” catalog controls all the dashboards features enabled, such as which tabs are displayed, which charts are enabled and which resources must be included. The following sections provide details about every configuration option.

7.3 Customization Guidelines

7.3.1 Dashboard Top Header

Change the BAM dashboards' top header following these steps:

1. Create a screen which implements the desired custom header (for example containing a custom logo with custom links).
2. Disable the default header by setting the "conf" column to "false" for the "defaultHeaderEnabled" entry in the "BAM_DefaultFeatures" catalog.



	id	conf
1	tabPerformanceMetricsEnabled	true
2	tabInstancesOverviewEnabled	true
3	tabActivitiesOverviewEnabled	true
4	tabUserActivitiesOverviewEnabled	true
5	tabCustom1Enabled	false
6	tabCustom1Label	Custom1
7	tabCustom2Enabled	false
8	tabCustom2Label	Custom2
9	tabCustom3Enabled	false
10	tabCustom3Label	Custom3
11	cssResourceId	BAM_Styles
12	defaultHeaderEnabled	false
13	additionalChart1Enabled	false
14	additionalChart1Label	Chart 1
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart 2

fig 38: BAM_DefaultFeatures catalog configuration for disabling the default header

As a result, the default header will not be displayed in the BAM Dashboards.

3. Include the newly created screen in the "BAM – Dashboards" screen inside the "CUSTOM_HEADER" template area. The new screen will be displayed instead of the default one.



fig 39: Inclusion of a custom header screen in the "BAM – Dashboards" screen

7.3.2 Dashboard CSS

The CSS styles included by default in BAM dashboards can be changed following these steps:

1. Duplicate the CSS resource with the id "BAM_Styles" and give it a new arbitrary id (e.g. "Custom_BAM_Styles").
2. Change the "conf" column for the "cssResourceId" entry in the "BAM_DefaultFeatures" catalog to match the newly chosen id.

	id	conf
1	tabPerformanceMetricsEnabled	true
2	tabInstancesOverviewEnabled	true
3	tabActivitiesOverviewEnabled	true
4	tabUserActivitiesOverviewEnabled	true
5	tabCustom1Enabled	false
6	tabCustom1Label	Custom1
7	tabCustom2Enabled	false
8	tabCustom2Label	Custom2
9	tabCustom3Enabled	false
10	tabCustom3Label	Custom3
11	cssResourceId	Custom_BAM_Styles
12	defaultHeaderEnabled	true
13	additionalChart1Enabled	false
14	additionalChart1Label	Chart 1
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart 2

fig 40: BAM_DefaultFeatures catalog configuration for changing the CSS resource

3. Modify the new CSS resource to achieve the desired dashboard look and feel. This is a standard CSS file, so standard styling techniques can be applied.

7.3.3 New Performance Metrics Charts

Up to four new charts can be added to the Performance Metrics dashboard. Follow the steps below:

1. Create a new screen that will host the chart and give it an arbitrary id (e.g. “BAMIncludeCustomChart”).
2. Enable the first custom chart by setting the “conf” column for the “additionalChart1Enabled” entry to “true” in the “BAM_DefaultFeatures” catalog.
3. Change the “conf” column for the “additionalChart1Label” entry in the “BAM_DefaultFeatures” catalog to set the desired chart title.

	id	conf
1	tabPerformanceMetricsEnabled	true
2	tabInstancesOverviewEnabled	true
3	tabActivitiesOverviewEnabled	true
4	tabUserActivitiesOverviewEnabled	true
5	tabCustom1Enabled	false
6	tabCustom1Label	Custom1
7	tabCustom2Enabled	false
8	tabCustom2Label	Custom2
9	tabCustom3Enabled	false
10	tabCustom3Label	Custom3
11	cssResourceId	BAM_Styles
12	defaultHeaderEnabled	true
13	additionalChart1Enabled	true
14	additionalChart1Label	My Custom Chart 1
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart 2

fig 41: BAM_DefaultFeatures catalog configuration for enabling a custom chart

4. Include the custom chart screen in the “BAM – Dashboards” screen inside the template area “ADDITIONAL_CHART1”.

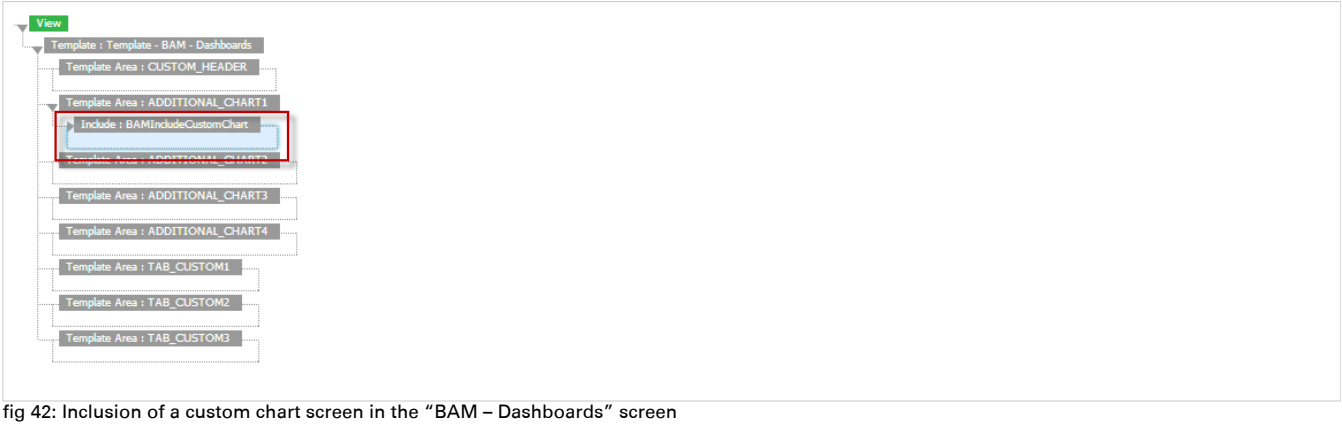


fig 42: Inclusion of a custom chart screen in the "BAM – Dashboards" screen

5. Modify the custom chart screen in order to implement the desired chart. It is possible to inherit the variable context from the "BAM – Dashboards" screen to make the chart dependent on user selections. For example, it is possible to inherit the "filter" variable of type "BAMFilter" to access the current filter selection:

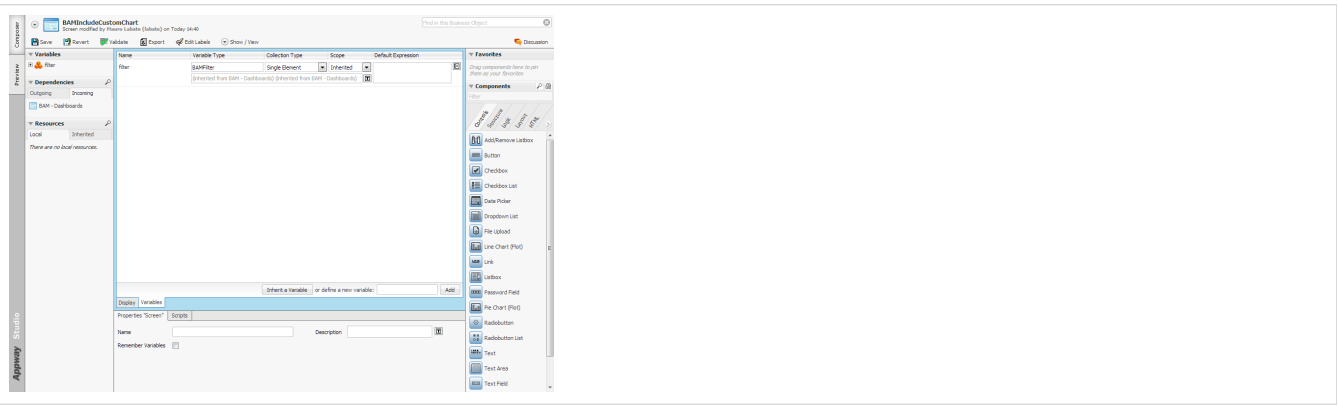
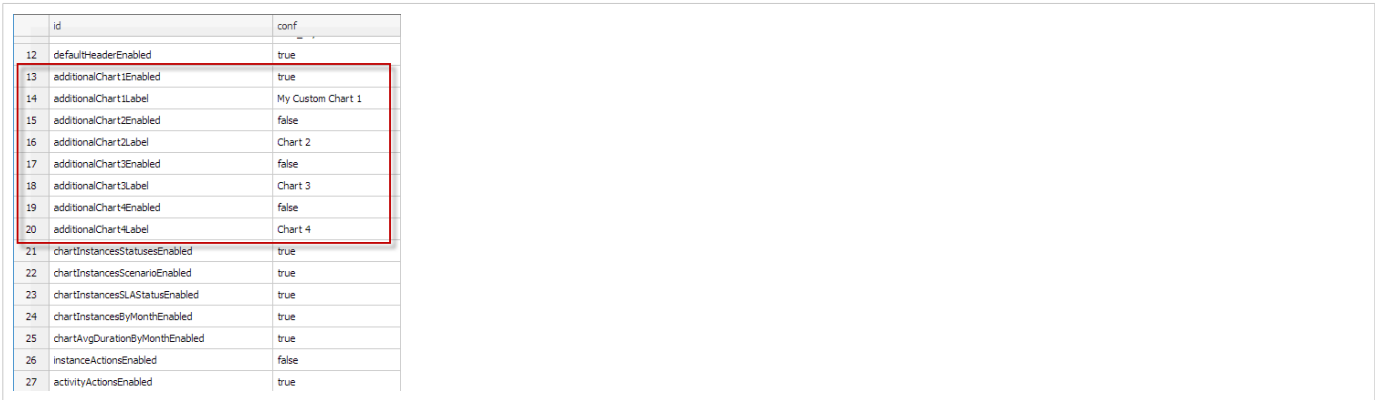


fig 43: Inheritance of the "filter" variable in the custom chart screen

The new chart can reuse one of the existing BAM Extension functions and just modify the data formatting/presentation, or it can even be implemented using any arbitrary logic (e.g. a custom query on the BAM database).

Up to four charts can be added following the aforementioned procedure, but using a different catalog entry id number. For example, for the second chart the properties to be updated are "additionalChart2Enabled" and "additionalChart2Label", and so on.



	id	conf
12	defaultHeaderEnabled	true
13	additionalChart1Enabled	true
14	additionalChart1Label	My Custom Chart 1
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart 2
17	additionalChart3Enabled	false
18	additionalChart3Label	Chart 3
19	additionalChart4Enabled	false
20	additionalChart4Label	Chart 4
21	chartInstancesStatusesEnabled	true
22	chartInstancesScenarioEnabled	true
23	chartInstancesSLAStatusEnabled	true
24	chartInstancesByMonthEnabled	true
25	chartAvgDurationByMonthEnabled	true
26	instanceActionsEnabled	false
27	activityActionsEnabled	true

fig 44: BAM_DefaultFeatures catalog configuration for enabling/disabling additional custom charts

7.3.4 Remove Out-of-the-Box Performance Metrics Charts

In order to remove one or more of the out-of-the-box charts in the Performance Metrics tab, update the "BAM_DefaultFeatures" catalog by setting the "conf" column corresponding to the chart to be removed to "false". In the picture below, the charts are matched with the corresponding properties to be enabled/disabled.

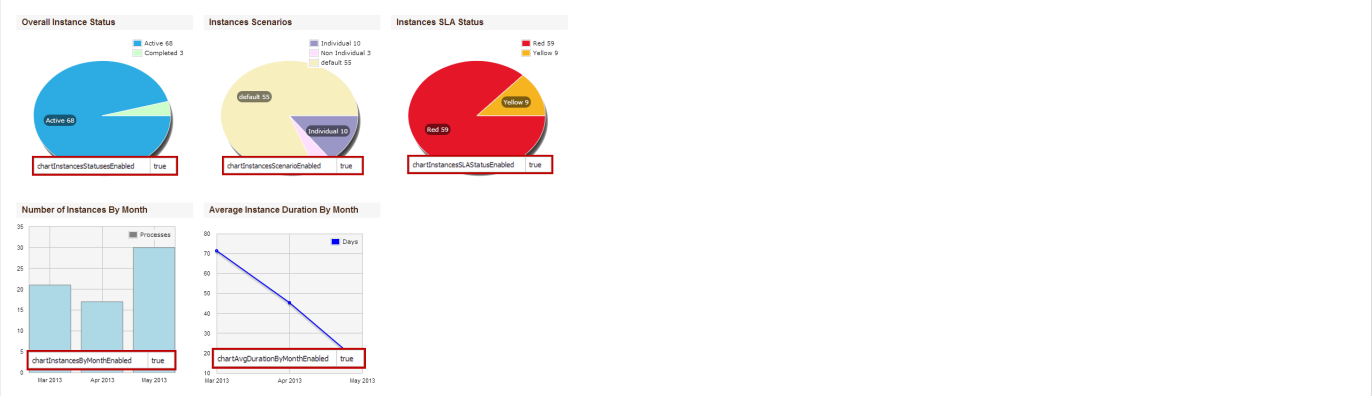
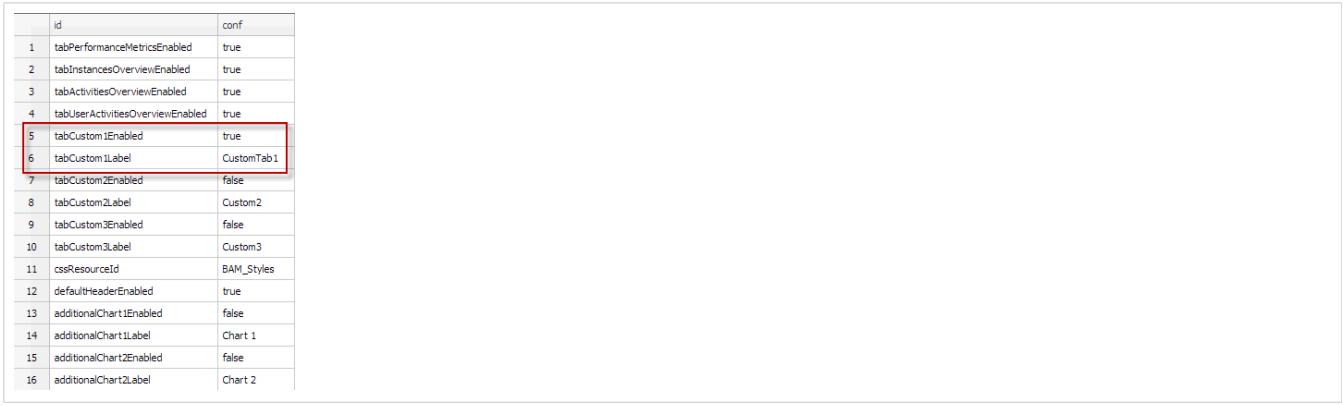


fig 45: Correspondance of BAM default charts with BAM_DefaultFeatures catalog entries to be enabled/disabled

7.3.5 New Dashboard Tab

Add up to three new dashboard tabs by following these steps:

1. Create a new screen that will host the content of the new tab and give it an arbitrary id (e.g. "BAMIncludeCustomTab").
2. Enable the first custom tab by setting the "conf" column for the "tabCustom1Enabled" entry to "true" in the "BAM_DefaultFeatures" catalog.
3. Change the "conf" column for the "tabCustom1Label" entry in the "BAM_DefaultFeatures" catalog to set the desired tab title.



	id	conf
1	tabPerformanceMetricsEnabled	true
2	tabInstancesOverviewEnabled	true
3	tabActivitiesOverviewEnabled	true
4	tabUserActivitiesOverviewEnabled	true
5	tabCustom1Enabled	true
6	tabCustom1Label	CustomTab1
7	tabCustom2Enabled	false
8	tabCustom2Label	Custom2
9	tabCustom3Enabled	false
10	tabCustom3Label	Custom3
11	cssResourceId	BAM_Styles
12	defaultHeaderEnabled	true
13	additionalChart1Enabled	false
14	additionalChart1Label	Chart 1
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart 2

fig 46: BAM_DefaultFeatures catalog configuration for enabling a custom tab

4. Include the custom tab screen in the "BAM – Dashboards" screen inside the template area "TAB_CUSTOM1".

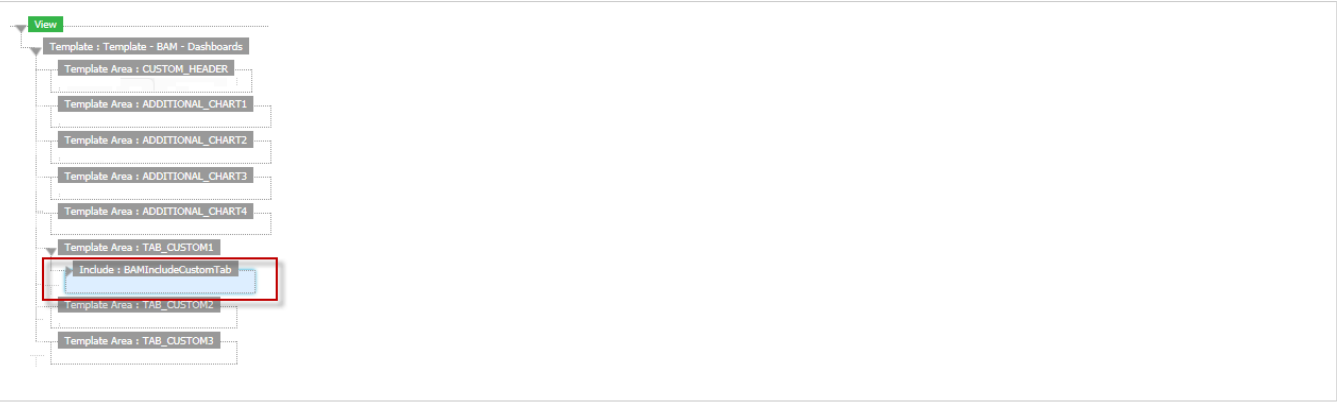


fig 47: Inclusion of a custom tab screen in the "BAM – Dashboards" screen

5. Modify the custom tab screen in order to implement the desired features. It is possible to inherit the variable context from the "BAM – Dashboards" screen to make the chart dependent on user selections. For example, it is possible to inherit the BAMFilter object to access the current filter selection:

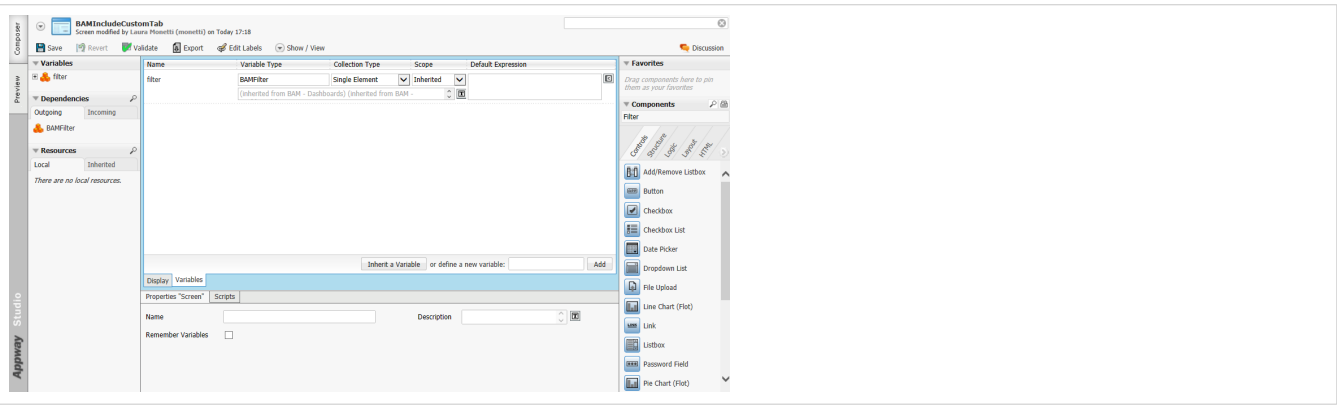


fig 48: Inheritance of the "filter" variable in the custom tab screen

The new tab can reuse existing BAM Extension functions to obtain the desired dataset and just modify the data formatting/presentation, or it can even be implemented using any arbitrary logic (e.g. a custom query on the BAM database).

Up to three tabs can be added following the procedure above, but using a different catalog entry id number. For example, for the second tab the properties to be updated are "tabCustom2Enabled" and "tabCustom2Label", and so on.

	id	conf
1	tabPerformanceMetricsEnabled	true
2	tabInstancesOverviewEnabled	true
3	tabActivitiesOverviewEnabled	true
4	tabUserActivitiesOverviewEnabled	true
5	tabCustom1Enabled	true
6	tabCustom1Label	CustomTab1
7	tabCustom2Enabled	false
8	tabCustom2Label	Custom2
9	tabCustom3Enabled	false
10	tabCustom3Label	Custom3
11	cssResourceId	BAM_Styles
12	defaultHeaderEnabled	true
13	additionalChart1Enabled	false
14	additionalChart1Label	Chart 1
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart 2

fig 49: BAM_DefaultFeatures catalog configuration for enabling/disabling additional custom tabs

7.3.6 Remove Out-of-the-Box Tabs

In order to remove one or more of the out-of-the-box dashboard tabs, update the "BAM_DefaultFeatures" catalog by setting the "conf" column corresponding to the tab to be removed to "false". In the picture below, the tabs are matched with the corresponding properties to be enabled/disabled.

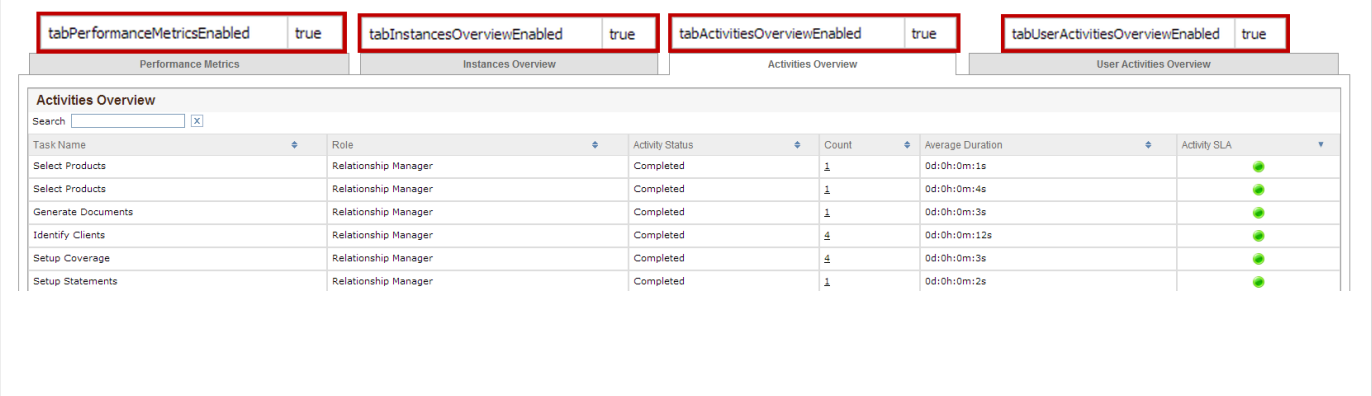


fig 50: Correspondance of BAM default tabs with BAM_DefaultFeatures catalog entries to be enabled/disabled

7.3.7 Custom Actions Column in the "Instances Overview" Tab

In the Instances Overview tab, it is possible to enable a special "Actions" column to implement custom actions to be executed on one of the instances displayed in the table.

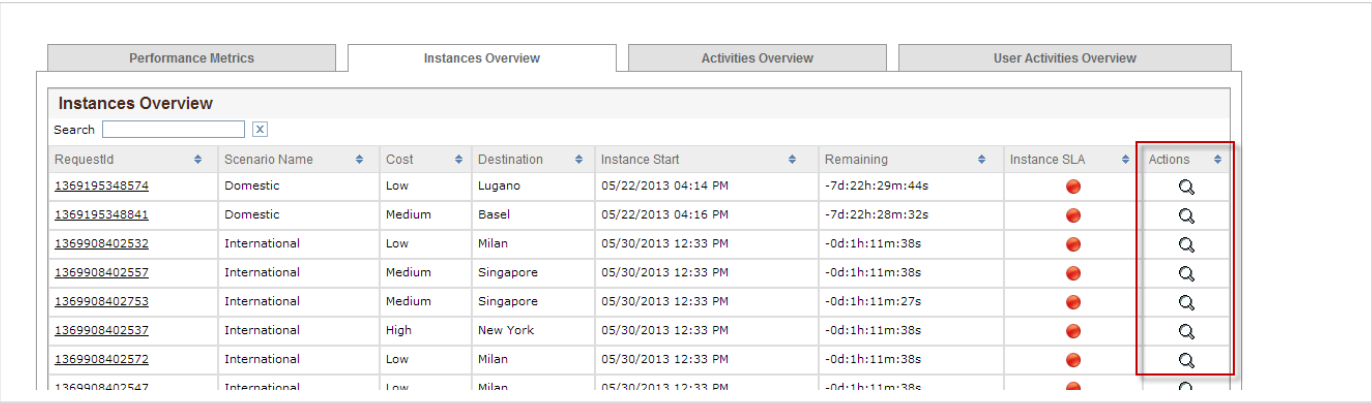


fig 51: Additional column "Actions" displayed inside the Instances Overview tab

The steps below describe how to defining an "Actions" column in the "Instances Overview" tab:

1. Enable instance overview actions by setting the "conf" column for the "instanceActionsEnabled" entry to "true" in the "BAM_DefaultFeatures" catalog.

	id	conf
13	additionalChart1Enabled	false
14	additionalChart1Label	Chart
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart
17	additionalChart3Enabled	false
18	additionalChart3Label	Chart
19	additionalChart4Enabled	false
20	additionalChart4Label	Chart
21	chartInstancesStatusesEnabled	true
22	chartInstancesScenarioEnabled	true
23	chartInstancesSLAStatusEnabled	true
24	chartInstancesByMonthEnabled	true
25	chartAvgDurationByMonthEnabled	true
26	instanceActionsEnabled	true
27	activityActionsEnabled	false

fig 52: BAM_DefaultFeatures catalog configuration for enabling the "Actions" column in the Instances Overview tab

2. Customize the "BAM – Include – Instance Actions" screen according to project-specific needs. By default, it already contains a sample "view" action which opens the details of the corresponding instance. This sample action can be safely removed and is merely provided as an example of how to implement a custom action.

   In the "BAM – Include – Instance Actions" screen, two variables are already defined: the "filter" (of type BAMFilter) which can be used to access the currently selected filters, and the "dbRecord" which is a named collection of strings containing the current process instance data. The "BAM – Include – Instance Actions" screen is included in the "ForEach" loop of the "BAM – Include – Instances Overview" screen as shown below:

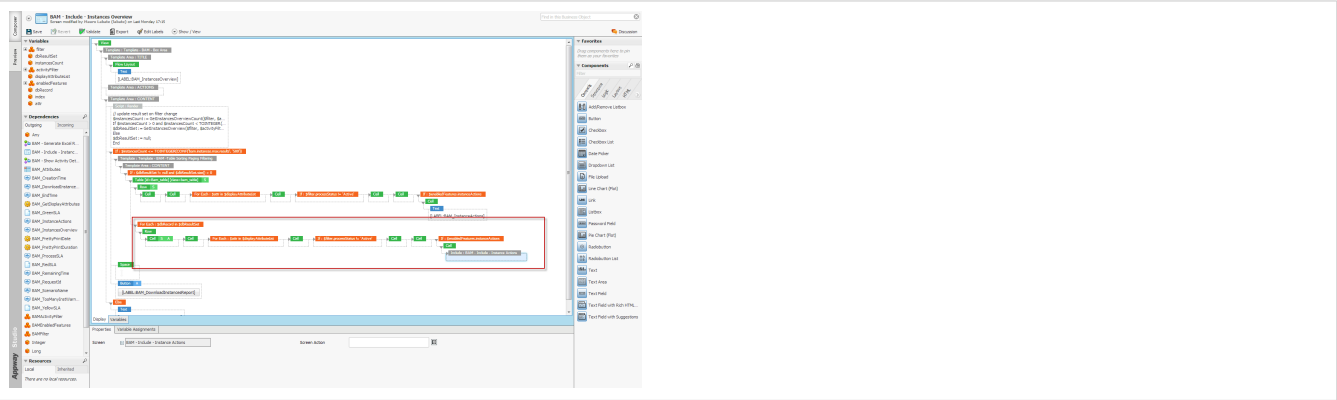


fig 53: "BAM – Include – Instances Overview" screen and "ForEach" loop on instances that contain the "BAM – Include – Instance Actions" screen

   Therefore, this included screen is rendered once for each row in the Instances Overview tab (i.e. once for each process instance). The variable "dbRecord" contains one of the records returned by the function "GetInstancesOverview" documented in the "Functions" chapter above.

Even though the actions screen can be fully customized, it is recommended to include only a set of icons, each representing a specific action on the instance. For example "view", "reassign", "delete", "notify", etc...

Using other type of components such as checkboxes or other controls is not recommended as part of the supported customizations.

It is the developer's responsibility to implement all the required checks in the custom actions screen in order to make sure that each action is only available if the instance is in an appropriate status. For example, the "reassign" action makes sense only for instances with the status "Active".

7.3.8 Custom Actions Column in the "Activity Details" Screen

Enable a special "Actions" column for the "Activity Details" screen in order to implement custom actions to be executed on one of the activities displayed in the table.

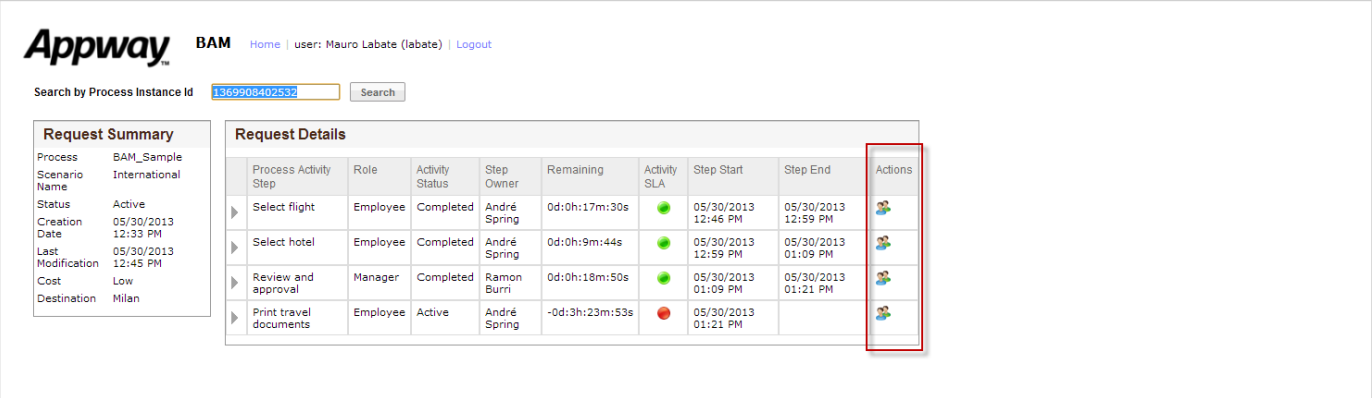
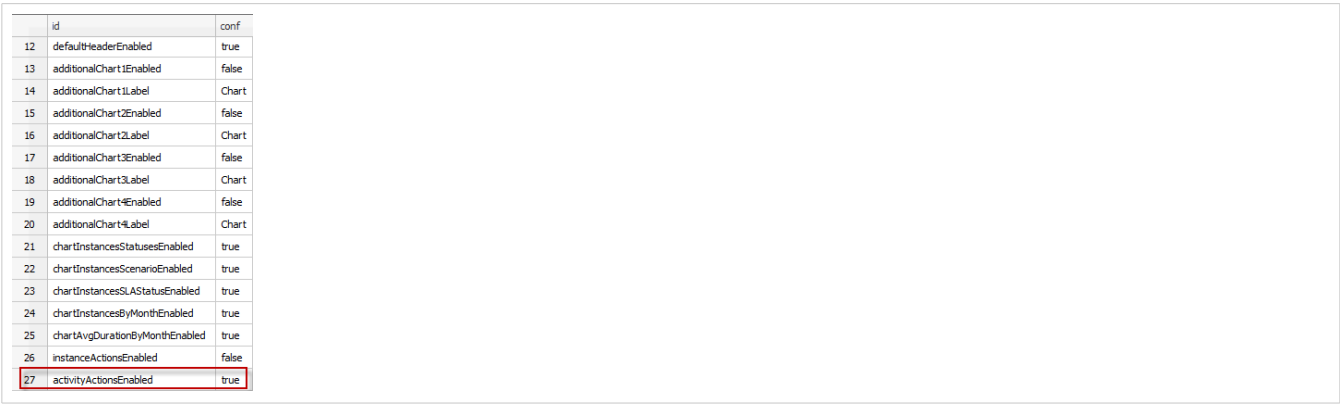


fig 54: Additional "Actions" column displayed inside the Activity Details screen

Define an "Actions" column for the "Activity Details" screen by following these steps:

1. Enable the activity details actions by setting the "conf" column for the "activityActionsEnabled" entry to "true" in the "BAM_DefaultFeatures" catalog.



	id	conf
12	defaultHeaderEnabled	true
13	additionalChart1Enabled	false
14	additionalChart1Label	Chart
15	additionalChart2Enabled	false
16	additionalChart2Label	Chart
17	additionalChart3Enabled	false
18	additionalChart3Label	Chart
19	additionalChart4Enabled	false
20	additionalChart4Label	Chart
21	chartInstancesStatusesEnabled	true
22	chartInstancesScenarioEnabled	true
23	chartInstancesSLAStatusEnabled	true
24	chartInstancesByMonthEnabled	true
25	chartAvgDurationByMonthEnabled	true
26	instanceActionsEnabled	false
27	activityActionsEnabled	true

fig 55: BAM_DefaultFeatures catalog configuration for enabling the "Actions" column in the Activity Details screen

2. Customize the "BAM - Include - Activity Actions" screen according to project-specific needs. In the "BAM – Include – Activity Actions" screen, one variable is already defined: the "dbRecord" is a named collection of strings containing the current activity data. The "BAM – Include – Activity Actions" screen is included in the "ForEach" loop of the "BAM – Include – Activity Details" screen as shown below:

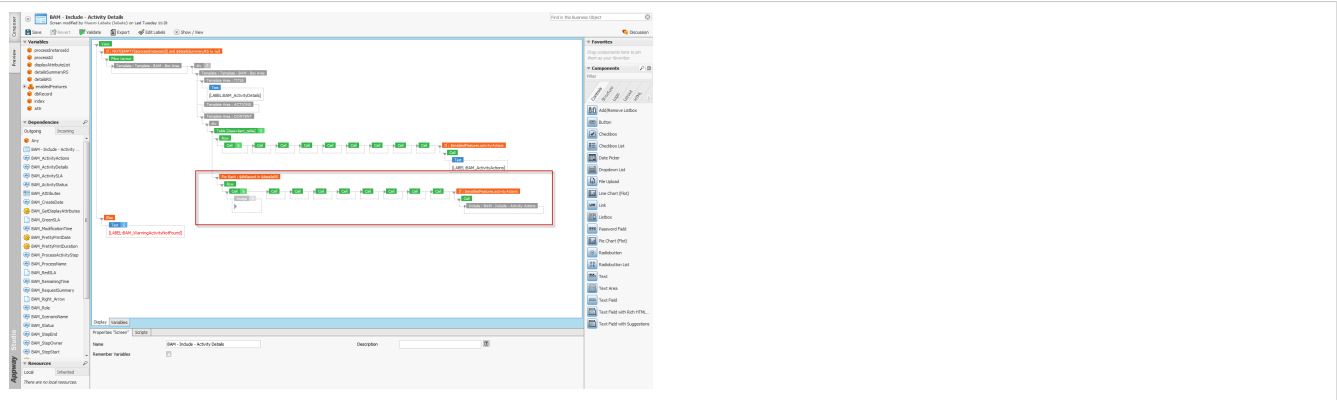


fig 56: “BAM – Include – Activity Details” screen and “ForEach” loop on activities that contain the “BAM – Include – Activity Actions” screen

Therefore, this included screen is rendered once for each row in the “Activity Details” screen activity table (i.e. once for each activity in the process instance). The variable “dbRecord” contains one of the records returned by the function “GetInstanceActivities” documented in the “Functions” chapter above.

Even though the “actions” screen can be fully customized, it is recommended to include only a set of icons, each representing a specific action on the activity. For example “view”, “reassign”, “delete”, “notify”, etc. Using other type of components such as checkboxes or other controls is not recommended as part of the supported customizations.

It is the developer’s responsibility to implement all the required checks in the custom actions screen to make sure that each action is only available if the activity is in an appropriate status. For example the “reassign” action makes sense only for activities with the status “Active”.

7.4 Customization Examples

Some implementation examples for the customization techniques described in the previous section can be found in the Appway Developer Cookbook.

Here are links to the recipes available:

- BAM Dashboard Customization: Add a New Chart (http://developer.appway.com/screen/ShowSingleRecipe/selectedRecipeId/1410808060792)
- BAM Dashboard Customization: Logo and Style Changes (https://developer.appway.com/screen/ShowSingleRecipe/selectedRecipeId/1410851131667)
- BAM Dashboard Customization: Add a Custom Action (https://developer.appway.com/screen/ShowSingleRecipe/selectedRecipeId/1412173277400)
- BAM Dashboard Customization: Send Notification upon SLA Expiration (https://developer.appway.com/screen/ShowSingleRecipe/selectedRecipeId/1411998873699)