Search documentation Search

Appian

Project Implementation Best Practices Checklist

Appian 7.7

NOTE: This site does not include documentation on the latest release of Appian. Please upgrade to benefit from our newest features. For more information on the latest release of Appian, please see the Appian 7.8 documentation

The information on this page is provided by the Appian Center of Excellence

Toggle Na

This page is *not* supported by Appian Technical Support. For additional assistance, please contact your Appian Account Executive to engage with Appian Professional Services.

Project Implementation Best Practices Checklist

These project implementation best practices allow teams to identify implementation risks during project development.

The following content can also be downloaded as a Microsoft Word document for sharing with teams and for offline completion: Project Implementatior Best Practices Checklist.docx.

User Experience

Item	**Importance**	**Status**
Application uses Tempo interfaces for all end user interactions (News, Tasks, Actions, Records and Reports)	Medium	
News events place key names/unique identifiers/statuses in the Post text	Low	
All actions have useful names and descriptions	Low	
No performance concerns from the team or client	High	
Application performance is stable or improving	Medium	
No user acceptance concerns from the team or client	High	
User acceptance is stable or improving	Medium	

Application Architecture and Design

Platform

Item	**Importance**	**Status**
A unique PREFIX and an object naming convention has been defined and followed for all objects	Medium	
Use a data centric design. The data is the most important piece of the application. Processes, rules and UI are means to view and act on the data.	High	
Persist all relevant business data into a database using data stores, query rules and query functions. Do not use the Query RDBMS smart service as the SQL statements do not get updated automatically when the entities are changed.	High	
Design short-lived processes to perform actions and maintain the data. Consider using process model based related actions instead of quick task related actions when possible.	High	
Use the Appian business database as the system of record for Appian. Ensure that any temporary exceptions to this are clearly communicated across development teams.	High	

Applications

Item	Importance	Status
All applications have security defined. In Tempo, this controls access to Actions and Feeds	Medium	
The structure of applications follows the Basic Application Assembly Procedure	Low	
Public applications are given intuitive names. They are displayed to users for filtering Actions by application	Medium	

Groups

Item	Importance	Status
All groups have security defined (for example. the administrators have been defined)	Low	
All groups are created as Custom groups	Low	
The "PREFIX Viewers" group for each application has been created	Medium	
The "PREFIX Designers" group for each application has been created	Low	
The "PREFIX Administrators" group for each application has been created	Medium	
All groups have as an Administrator member the "PREFIX Administrators" group	Medium	
After deployment to production, do not delete groups. The group identifiers are reused when groups are deleted which can affect the security and behavior of an application.	Low	

Data Types

Item	Importance	Status
Number of CDTs is known	Low	
All CDT names use the form PREFIX_camelCase	Low	
All CDT namespaces have been changed to follow urn: pattern	Low	
All CDTs stored in a data store expose a primary key field	Medium	
All CDTs contain no more than 50 fields	Low	
All CDTs contain no more than 1 level of nested CDTs	Low	
No CDT contains nested lists that are not explicitly defined as separate CDTs - e.g. a nested list of Text	Medium	
After deployment to production, do not rename, delete or change the type of existing CDT fields. Making these non-backward compatible changes to a CDT may break process instances and rules.	High	
When adding new fields to a CDT already in used, ensure that the process models and rules using this CDT can properly handle null values in the new CDT fields.	High	

Data Stores

Item	Importance	Status
All data stores have security defined	Medium	
All data stores follow the naming convention "PREFIX "	Low	
All Data stores do not mix CDTs mapped to DB Tables and CDTs mapped to DB Views, for example, DB Views should have their own data stores	Low	

Data Type Schema

Item	Importance	Status
Historical data is stored in its own table or schema	Medium	
If flat CDTs have fields that store IDs of other records, the CDTs should have foreign key constraints defined in the database schema	Low	
The resource utilization metrics of the most used SQL statements have been calculated	Medium	
The DDL SQL for all tables and schemas is stored in version control and used to create tables and schemas needed to publish Data Stores.	Low	

Database Views

Item	Importance	Status
Views are used sparingly and only where appropriate	Medium	
The EXPLAIN statement has been run on all views and the results have been analyzed	Medium	

Tempo Record Types

Item	Importance	Status
All Record Types have a description	Low	
All Record Types have a custom URL Stub instead of the auto-generated URL Stub	Medium	
All Record Types have security defined using Record specific groups	Medium	
All Record Types should have security defined for at least Administrators and Viewers	Low	
All List Views and Dashboards are defined in rules (and not directly in the Expression fields)	Low	
Entity backed records are secured using group id(s) stored in the database	Low	

Tempo Records Dashboards

Item	Importance	Status
All dashboards do not display more than 50 fields	Medium	
All dashboards do not call more than 3 query rules	Medium	
All dashboards use the function load() when displaying paging grids and dynamic behaviors	Low	
Multiple Record Dashboards are used to show related data that is not a record in itself	Low	
For view-only purposes and to drill-into details, use dynamic behavior (SAIL) on dashboards instead of Related Actions	Low	
All List Views and Dashboards have been tested for performance	Medium	

Process Model Folders

Item	Importance	Status
All folders have security defined	Low	
All models have descriptions	Low	
All model names use the form "Verb Noun"	Low	

Process Model Properties

Item	Importance	Status
All models have security defined at the lowest security level possible	Medium	
All models use dynamic display names	Low	
All models define the shortest archiving period possible	Low	
All process variable names use the form camelCase	Low	
After deployment to production, do not add required parameters to existing process models as this would prevent running parent processes from calling these process models as a sub-processes. When new non-required parameters are added, make sure that the process model can handle the use case where these new parameters are empty/null at runtime.	High	
All models have Alert settings specified e.g. the alerts should be sent to application administrators group	Low	

Process Model Diagram

Item	Importance	Status
Process instance security is set when necessary	Medium	
Process models are split into sub-processes to compartmentalize sets of functionality and large cumbersome process models are avoided	Medium	
Horizontal swimlanes are used in models with attended tasks	Low	
Swimlane task assignment is used only where necessary and utilizes groups, rules, or process variables for assignment	Low	
Unattended nodes are assigned to execute in the context of the Designer instead of the Initiator	Medium	
Models contain no more than 30 nodes	Medium	
Models contain no more than 50 process variables	Medium	
All node names use the form "Verb Noun"	Low	
All XOR/OR gateways have a single incoming flow	Low	
All gateway names are in the form of a question	Low	
All outgoing flows from a gateway are labeled	Low	
XOR gateways are used instead of OR	Low	
XOR gateways are used in front of MNI nodes to check for empty/null values	Low	
Process flow will always reach at least one terminating end event	Low	
Process-to-process messages are targeted to a specific process instance using PID (except when starting a new process)	Medium	
All complex logic is documented using annotations (anything that isn't obvious)	Low	
All external integrations are contained in their own subprocesses to minimize the impacts of the external systems changing their interfaces. E.g. anything other than query rules and data stores should be encapsulated.	Low	
If the external integration points are using CDTs to exchange data (such as integration with web services), use these CDTs locally within the integration process models or rules and create business CDTs to be used by the rest of the application. This prevents changes in external systems data structures from having widespread impacts in the application.	Low	
Query rules and Write to Data Store nodes are used instead of the Query Database node	Low	

Use the Secure Credentials Store instead of plain username/pwd	Medium	
Business data and SLAs are persisted into the business database	Medium	
Best practices for created memory efficient models have been followed		
Use constants to reference documents that are used in the process nodes. For example, use a constant to reference a document template used in the Generate Text Document smart service.	Medium	

Process Node Properties

Item	**Importance**	**Status**
All nodes have descriptions	Low	
Attended tasks use dynamic display names	Low	
Attended tasks use swimlane assignment	Low	
Node inputs do not make the same query rule call more than once	Medium	
All expressions use isnull() or length() to check for null/empty parameter values, for example, the expression checks that the parameters are not null prior to using them	Low	
CDTs are not passed by reference between parent and sub-process	High	
Looping functions are used instead of Multiple Node Instances where possible	Medium	
Start nodes do not allow users to step back from the next chained activity	Medium	
Sub process nodes do not allow users to step back from the next chained activity	Medium	
Smart services that perform an action (create or update DB, document, etc...) do not allow the users to step back from the next chained activity	Medium	
Use activity class parameters instead of process variables to store intermediate data used on a node or form that is not needed by the rest of the process	Medium	
On the Other tab for Forms, check the "Delete previous instances" and do not check "Keep a record of the form"	Medium	
Use groups for task assignment and security settings	Medium	
Use rules and constants instead of hard-coded values in the process nodes	Medium	

Task Forms

Item	**Importance**	**Status**
User forms are written in SAIL and SAIL expressions are invoked with keyword syntax.	Medium	
All forms contain fewer than 20 inputs	Low	
Query rules are executed and stored in ACPs and not directly on Form element default values	Medium	
All nodes contain fewer than 3 query rules	Medium	
After deployment to production, do not add new inputs to the form SAIL expressions. When new inputs/outputs are required, create new SAIL expressions to be used by the new version of the process models.	High	

Rules and Constants

Item	**Importance**	**Status**
The root folder securities do not grant Editor privileges to all users (this is the default setting that must be changed)	Medium	

All folders have security defined or inherited	Low	
All rules are organized into application specific folders	Low	
All Constant names use the form PREFIX_ALL_CAPS	Low	
All Rule names use the form PREFIX_camelCase()	Low	
All Query rule names use the form PREFIX_getCdtTypeByFilterFields()	Low	
All rules have descriptions that include the returned data type	Low	
All recursive rules will never exceed a total depth of 20	Low	
All complex rules use comments /**/ to describe complex logic	Low	
Always use keyword syntax with parameter names when invoking the rules.	High	
After deployment to production, do not add inputs to existing rules unless all invocations of the rule always use keyword syntax with parameter names.	High	
After deployment to production, do not change the names of rule parameters that may not match how parameter names are set up in previous versions/releases.	High	
After deployment to production, do not delete rules or constants. Instead deprecate them using the following guidelines: - Deprecation means adding a description or comment that prevents designers from invoking the same rule. - Deprecation comment should include a standard string used across all objects (e.g. "[DEPRECATED]") and a comment informing designer of replacement object to use, if applicable.	High	

Document Folders

Item	Importance	Status
Application specific contents are stored within an application specific Community and folder structure (or at least not in the Default Community)	Low	
All communities/KCs/folders have security defined	Medium	
All reports are stored in the application folder structure (not in the System Reports folder)	Low	
KC and folders created in process have specific security defined when required	Low	

Legacy Portal Reports

Item	Importance	Status
All reports can use sorting/filtering to display relevant rows without paging	Low	
All reports have been measured and optimized for performance	Medium	

Tempo Reports

Item	Importance	Status
All Tempo Reports have descriptions	Low	
All Tempo Reports have custom URL stubs	Low	
All Tempo Reports have security defined using specific groups	Medium	
All Tempo Reports have security defined for at least Administrators and Viewers	Low	
All Tempo Reports are defined in a rule (and not directly in the Expression field)	Low	
All Tempo Reports do not contain more than 3 grids or diagrams	Medium	
All Tempo Reports have been tested for performance	Medium	
All Tempo Reports use the function load() when displaying paging grids and dynamic	Low	

behaviors		

Feeds/News

Item	Importance	Status
All feeds have security defined	Medium	
News entries related to records contain the corresponding record tags	Medium	
When using record tags, the security settings on the corresponding records must match the news entry audience	Medium	
When using attachments, the security settings of the files must match the news entry audience	Medium	

Miscellaneous

Item	Importance	Status
Groups, rules, and/or constants are used instead of hardcoded usernames in security settings and other areas, such as Assignment Tab, for all application objects	High	
URLs are stored in Rules and Constants and not hardcoded in application objects	Low	
The signature of a smart service plugin module cannot be modified after the plugin has been deployed. Modifying the inputs or outputs of the smart service require creating a new smart service and reconfiguring the process models that use it.	Low	

Quality Assurance

Item	Importance	Status
Test scripts cover all major use cases	High	
Test scripts are executed regularly	Medium	
Test scripts are executed using the correct role-based test accounts (not sysadmin accounts)	High	
Automated test scripts have been built	Low	
Automated test scripts cover all major use cases	Low	
Automated test scripts are executed regularly	Low	
Custom plugin code has been peer-reviewed	Medium	

Change Control

General

Item	Importance	Status
Application export is tested regularly	Medium	
Application import is tested regularly	Medium	

Version Control

Item	Importance	Status
Repository includes application export package	Low	
Repository includes Appian Configuration Manager	Medium	
Repository includes SQL scripts for database schema creation/modification	Medium	
Repository includes custom plugin code	Medium	

Repository includes all configuration changes e.g. data sources, custom.properties, gateway, topology, etc...	Medium	

User Guides by Role

- Designer
- Developer
- Web Admin
- Server Admin (On-Premise Only)

Tutorials

- Records
- Interfaces
- Process

Release Information

- Release Notes
- Installation
- Migration
- System Requirements
- Hotfixes
- Release History

Other

- STAR Methodology
- Best Practices
- Glossary
- APIs