



Tricentis Tosca Standards & Best Practices

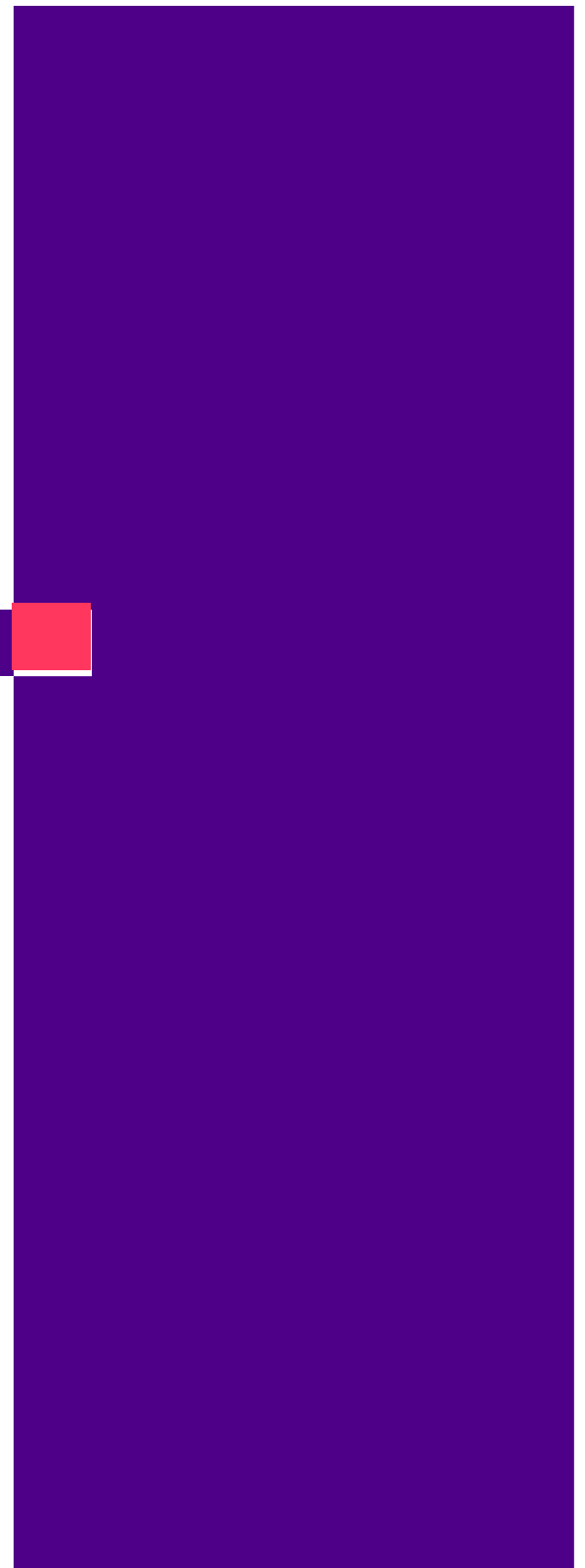


Table of Contents

1	Introduction	5
2	Tosca Artifacts	6
2.1	Explanation of Tosca Icons	6
2.2	Tosca Artifact Relationship	7
3	Workflow	8
3.1	Simple Workflow	8
3.2	Approval Workflow (4-Eyes Check)	8
3.3	Two-step approval	8
3.4	Folders for each tester	9
4	Requirement Section (Yellow).....	10
4.1	Existing Requirements Specifications.....	10
4.2	Modeling Requirements in Tosca	10
4.2.1	Requirement Set or Requirement	11
4.2.2	Level of detail for the requirement structure	11
4.2.3	Creating a Requirement Set from the scratch	12
4.2.4	Relationship between Requirement, TestSheet and TestCase	14
4.2.5	Weigh Requirements	17
4.2.6	Relative Weight and Contribution	18
5	TestCaseDesign Section (Red)	19
5.1	Introduction to TestCaseDesign	19
5.1.1	Advantages when using TestCaseDesign	19
5.1.2	General characteristics for good TestCaseDesign.....	19
5.1.3	The TestSheet.....	20
5.1.4	When to use TestCaseDesign – and when not to.....	20
5.1.5	How to get started.....	21
5.2	TestSheet - Best Practices.....	22
5.2.1	Basic TestSheet Structure.....	22
5.2.2	Generating Instances using Combinatorics.....	23
5.2.3	Using equivalence classes and specific values.....	23
5.2.4	Linking TestCases with TestCaseDesign	23
5.2.5	How to reduce maintenance effort further: Classes	24
6	Modules Section (Orange).....	26
6.1	Structure and Naming	26
6.1.1	Structure and Naming of Modules.....	26
6.1.2	Naming of ModuleAttributes.....	27
6.1.3	Breaking down Modules	28
6.1.4	Sequence of ModuleAttributes.....	29
6.1.5	Duplicate controls and cardinality.....	31
6.2	Creating Modules correctly	32
6.2.1	Scanning and Steering	32

6.3	Standard Modules.....	38
6.3.1	Purpose.....	38
6.3.2	When to use which Standard Module.....	38
6.4	Maintaining Modules.....	38
6.4.1	Using Rescan.....	38
6.4.2	Replacing Modules	38
6.4.3	Modifying Module information	39
6.4.4	Scrolling Behavior	40
6.4.5	User Simulation.....	41
7	TestCase Section (Blue)	42
7.1	Tips on TestCase Automation.....	42
7.1.1	Usage of TestCaseWorkState.....	42
7.1.2	Standard folder structure within a TestCase	42
7.1.3	Naming convention of TestCases.....	44
7.1.4	Naming convention of TestStep Folders.....	45
7.1.5	Naming convention of TestSteps.....	45
7.1.6	TestStepLibraries	46
7.1.7	Using fixed values.....	48
7.1.8	TestCase Templates	49
7.1.9	Recovery and CleanUp Strategy	49
7.1.10	Advanced verification	49
7.1.11	Cross Browser testing.....	51
7.1.12	Timeout and Wait.....	51
7.1.13	When to split TestCases	52
7.1.14	Conditions	53
7.1.15	Error Handling done right.....	54
7.1.16	Using Configuration Parameters	57
7.1.17	Project Settings	57
7.1.18	Using Tosca Recorder.....	58
7.1.19	API Scan.....	58
8	Execution Section (Green).....	59
8.1	Folder structure.....	59
8.2	ExecutionList naming convention:.....	59
8.3	Folder structure in an ExecutionList	59
8.4	Test Mandates	60
8.5	Distributed Execution.....	60
8.6	TC Shell.....	60
8.7	Exploratory testing	60
8.8	Interactive testing.....	60
9	Reports	61
9.1	Reporting	61

9.2	Requirement Set – Dashboard	61
9.3	Reporting Section	61
10	Best Practices	62
10.1	Do's and Don'ts	62
10.1.1	TestCase Objects	62
10.1.2	Modules	62
10.1.3	TestCaseDesign	63
10.1.4	TestCases	63
10.1.5	ExecutionList	64
11	Resources	66
11.1	Web Links	66

1 Introduction

This document provides the Tricentis Tosca standards, which should be applied to all Tosca projects. In addition, a set of best practices is included to help solve common challenges.

However, this documentation does not cover any Tosca training. Before reading, you should have a basic understanding of the Tosca objects. (See chapter 2.1)

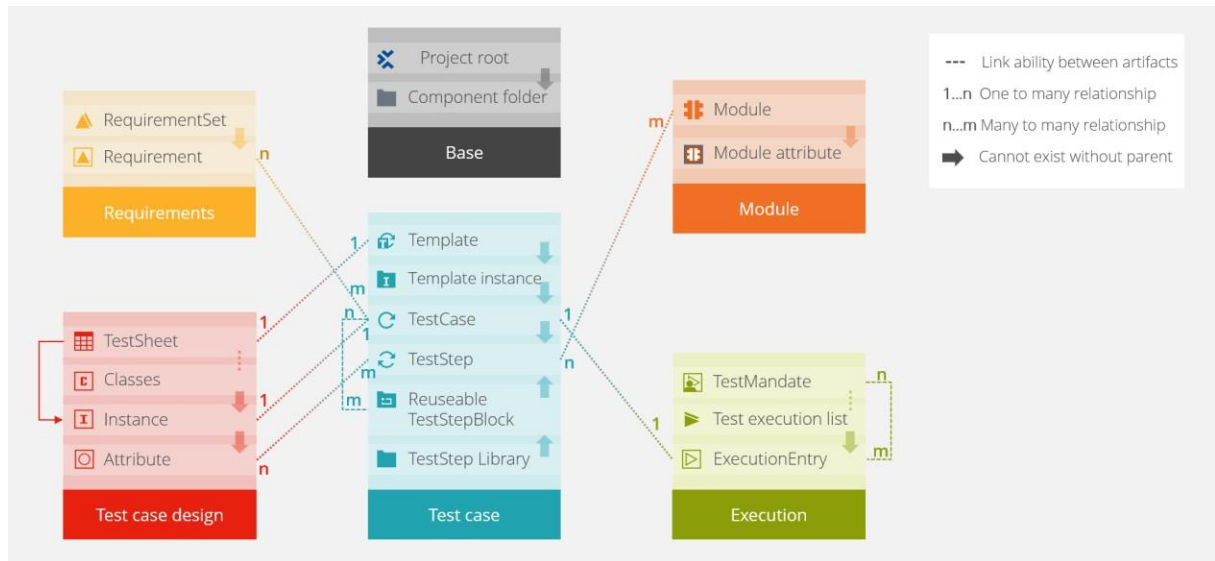
2 Tosca Artifacts

2.1 Explanation of Tosca Icons

The following artifacts have significant importance in Tosca:

	RequirementSet	A Requirement Set is a collection of related Requirements.
	Requirement	A Requirement in Tosca represents a specific Requirement in the System under Test. For easier Reporting and readability Requirements can be structured in other Requirements and therefore be collected on different levels.
	Test Data Sheet	A Test Sheet represents the test data for a specific Template. Test Cases are generated from each Instance of the Test Sheet.
	Data Attribute	Data parameter that will have multiple values that need to be varied to create different data scenarios.
	Data Instance	Data values of Data Attributes. Each different data value will be a Data Instance.
	Test Case	Test Cases contain a sequence of manual or automated Test Steps which need to be executed to test your system. Test Cases can also include so called Test Step Folders and statements to organize the test flow.
	Test Step	Test Steps are one or more actions to be done in the test. A Test Step has one or more Test Step Values included. One example could be a Login Test Step with 2 values "Username" and "Password". In automated Test Cases a Test Step is always created from a Module which usually represents one screen (e.g. Login). A Test Case is built by combining different Test Steps in a specific order.
	Test Step Block	A Test Step Block contains a sequence of centrally managed Test Steps that can be re-used in various test cases. The concept is equivalent to Test Case Design classes.
	Test Step Library	A Test Step Library is a centralized storage for Reusable Test Step Blocks. Modifying a Test Step within the library automatically updates all referenced Test Steps in the respective Test Cases. The concept of re-usability in a Test Step Libraries is like the concept of Test Sheet classes.
	Test Case Template	A Test Case Template is a generalized Test Case that utilizes a Test Sheet as data source. It works the same way as a serial letter where you have one master template to generate any similar letters with e.g. varying receivers. A Test Case Template, in most of the cases, focuses on a specific process, or a specific test-section. With Template-Conditions a Template can be made more generic, so that various test-paths can be taken, according to given input data.
	Test Case Template Instance	A Test Case Template Instance folder is a container, where Test Cases from a Test Case Template are generated with different data variations. For example, if a Template is made for testing the creation of customers, there will be Test Cases created via the Template for each customer type. This data for the different variants is stored in Test Case Design and linked to the Test Case Template as reference.
	Module	Scanned screen or NonUI Components for different applications: A Module represents one screen in an application (UI-Tests) or one Service/XML (NonUI-Tests). These Modules are simplified representations of a test object (e.g. web page) used in test automation. They are created by scanning the test object utilizing the Tosca XScan.
	Module Attribute	UI or non-UI element. Typically, a control (e.g. button, table, tree, dropdown, slider, field, ...), on non-UI technologies a message field (e.g. XML Element, JSON Array/Elements).
	Execution List	A collection of Test Cases, that can be meaningfully grouped together (e.g. Smoke Test, Regression, ...). The Test Case groups and the folder structure above the Execution List should be created according to Reporting requirements.
	Test Mandate	With Test Mandates, several testers can execute any parts of an Execution List independently and simultaneously. The results are automatically transferred to the result list of the Execution List (Actual Log) and consolidated.
	Business Execution List	Business Execution Lists are used to include and/or exclude results and to link them to Requirement Sets later. Business Execution Lists are not used for executing Test Cases.
	Defect	Tosca Commander categorizes undesired behavior issues as Defect. They can be created in the Issue section of Tosca.
	Feature Request	Tosca Commander categorizes missing or desired behavior issues as Feature Requests. They can be created in the Issue section of Tosca.

2.2 Tosca Artifact Relationship



3 Workflow

3.1 Simple Workflow

The basic workflow for building objects should be done in the following way. Each section may consist of multiple folders:

- Application
 - Contains the finished artifacts.
- _Templates
 - Samples to be used by everyone.

To separate the creation from the final artifacts, it is recommended to create a sandbox area for each user (e.g.: ZZ_Sandbox).

3.2 Approval Workflow (4-Eyes Check)

To enable an approval workflow, the folder structure should be done in the following way:

Each section has at least three folders:

1. Approved
 - ➔ Only to be used by the Test Approval user group.
2. In Review
 - Used by everybody. Once the object is ready to be approved.
3. Draft
 - ➔ Used by everybody. During the creation phase of each object.

In the Modules/TestCase/TestCaseDesign section is a special folder:

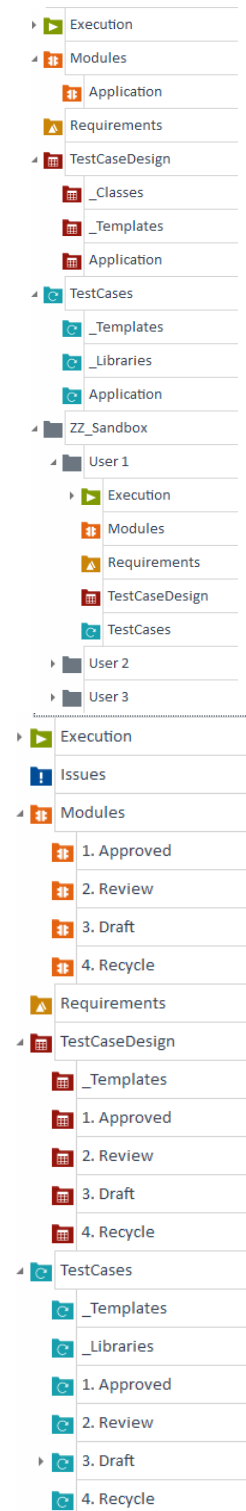
4. Recycle
 - ➔ If a module or test case could be used later, it is saved there.
 - ➔ Ensure to clean up the folder regularly.

3.3 Two-step approval

When test cases are in the "In Review" folder all test cases are approved in two steps:

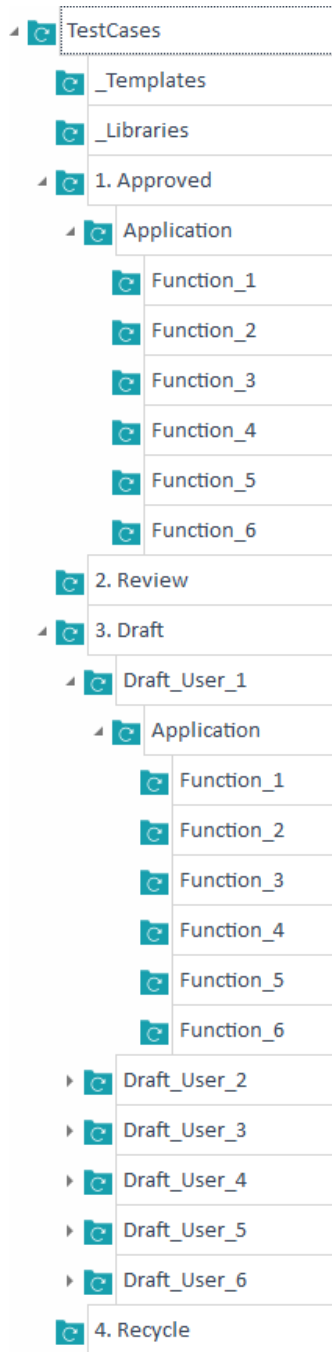
- 1) Best practice Tosca implementation check (reusable parts, naming, stability)
- 2) Business check and specification check (all specified required parts are done correctly) After

those checks, the test cases can be moved to the approved folder.



3.4 Folders for each tester

The folders "2_In Review", "3_Draft" and "4_Rejected" contain folders for each tester. This structure is different from the structure in the "1_Approved" folders.



However, each user folder has the same structure as the Approved folder.

4 Requirement Section (Yellow)

A requirement is a functional or non-functional criterion, which is relevant to the test project. Accordingly, the object type Requirement equally represents requirements and functional units.

Definition: *A requirement is a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation. Thus, a requirement represents a condition or capability needed by a user to solve a problem or achieve an objective.*

In Tosca, we build a requirement structure based on functional requirements and use this structure to weight the business risk of each requirement. Once the requirement structure has been created and weighted, TestCases can be linked to the Requirements. The requirement section can then be used to analyze test coverage based on the business risk, to identify which requirements haven't been covered by testing yet and to further plan the test strategy.

In general, Requirements can be grouped into 2 types:

- Regression – View recurring functionalities
- Progression – View newly developed functionalities

4.1 Existing Requirements Specifications

For a specific customer, most applications already have detailed requirements specifications on a platform. Those requirements form the basis of a risk analysis which is used to compile an adequate regression requirement structure in Tosca.

4.2 Modeling Requirements in Tosca

When modeling requirements in Tosca, there are a few rules that should be applied to get maintainable and usable requirement sets.

Structuring Requirements

Before creating a Requirement Set, ask yourself “What should be the basis for my reporting?”. Requirement Sets should be built redundancy-free and should have a clear structure.

- Requirement Sets themselves should be structured according to business objects to avoid having a long list of requirements.
- However, this structure should not be too deep to keep it clear and maintainable.
- There should be max. 8 to 10 requirements on one level, otherwise it becomes difficult to read and difficult to assign relative weights

Level of Specification

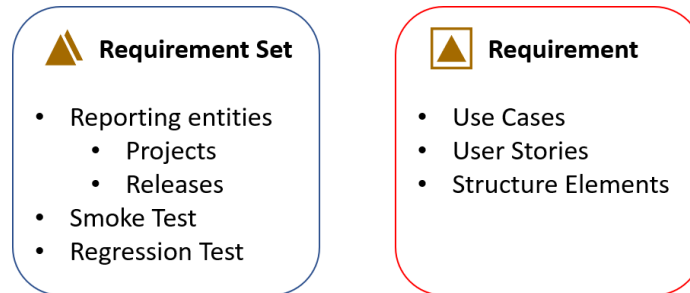
In the requirement section the system under test is modeled down to the level of use cases, user stories, but not further.

Establishing different views

All views onto the test portfolio, which are needed within the test project, are established in the Requirement Section.

4.2.1 Requirement Set or Requirement

At the top of every requirement structure in Tosca is the Requirement Set, which acts as a container for the requirement structure. The following pictures explain when to use Requirement Sets and when to create Requirements within the set.



4.2.2 Level of detail for the requirement structure

The necessary level of detail for the requirement structure depends on the application described, but the goal when creating the structure should always be to create a lean and easy to understand requirement structure with as little detail as possible without missing any major requirements.

For each requirement added to the Requirement Set, you should ask yourself whether it is necessary. In addition, think about how many TestCases should test this requirement. Because often, if only one TestCase is necessary for a requirement to verify it, the requirement is more likely only a data variation, an acceptance criterion or even only a TestStep or verification.

There is no strict rule to follow here, but the following list describes items that should be added as a requirement and items that should not.

Add as a Requirement to the Requirement Set:

- Major Business Requirements
 - e.g. Manage account, Search for customer, ...
- Processes
 - e.g. Process fraud, Transfer money, Create new account, ...
- Sub-Processes/Process Steps, if relevant
 - e.g. Enter customer data, check for existing account, create account, send notification, ...
- Business Entities that have an impact on the behavior of the application
 - Type of account, Application roles, Type of customer, ...
- Structure Elements to group the Requirements and Processes

Do not add to the Requirement Set:

- Business Entities that do not have an impact on the behavior of the application
 - e.g. Type of Credit Card, Country of customer, ...
 - Business Entities without impact are usually part of a TestCase
- Data Variations
 - e.g. Amount of transaction, Age of customer, Reason for fraud
 - Data Variations are usually a part of the TestSheet to create the different TestCases based on those variations. Only if such data variations have huge impact on the business functionality and behavior, they are added to the requirement structure.

- Conditions
 - e.g. when A then do B, when limit is reached, deny transaction, ...
 - Such detailed requirements are usually a part of a larger requirement and should be added as a part of the TestSheet or TestCase
- Acceptance Criteria
 - e.g. the field must be locked for Customer Type A, Show transaction confirmation
 - Acceptance Criteria are part of TestCases and are verified by those TestCases. Like Conditions, they should belong to a "bigger" requirement.

4.2.3 Creating a Requirement Set from the scratch

To create the requirement structure, first decide how to structure your requirements on the top level, the Requirement Sets. Often, one application is described in one Requirement Set. Later, different views can be created as separate Requirement Sets (like a Release or Sprint View, or a specific Requirement Set that contains all the requirements relevant for Regression Tests).

When creating the requirement structure, start thinking on the highest level:

	Name	Frequ...	Dama...	Weight
▲ ▲	Consumer and Business Banking			
▲	Customer			1
▲	Account			1

Once you have created the core business requirements, continue deepening these levels:

	Name	Frequ...	Dama...	Weight
▲ ▲	Consumer and Business Banking			
▲ ▲	Customer			1
▲	Creation			1
▲	Modification			1
▲	Closure			1
▲	Account			1

Try to keep the structure lean and simple. The main reason to create the requirement structure is to make risk-weight testing possible. Do not add details to the requirement structure that belong to a TestSheet or TestCase. Try to think simply at the beginning – an initial version of a requirement structure could look like this:

	Name	Frequ...	Dama...	Weight
▲	Consumer and Business Banking			
▲	Customer			1
	Creation			1
	Modification			1
	Closure			1
▲	Account			1
	Creation			1
	Modification			1
	Closure			1

Hint: When there are major differences between Requirements like there are between creation and closure of Customers and Accounts, these processes should be weighed separately.

Make sure that requirements on the same level do not represent areas that belong to each other (e.g. cards, credit cards, debit cards – as the last two might belong to the first):

False:

	Name
▲	Sample FALSE
	Cards
	Credit Cards
	Debit Cards

Correct:

	Name
▲	Sample CORRECT
▲	Cards
	Credit Cards
	Debit Cards

During the process of creating the requirement structure, you might recognize that there are topics, which can be either on the highest level or lower levels of several requirements:

	Name	Frequ...	Dama...	Weight
▲	Consumer and Business Banking			
▶	Customer			1
▶	Account			1
	Billing (Fees and Interest rates)			1

Name	Frequ...	Dama...	Weight
Consumer and Business Banking			
Customer			1
Creation			1
Modification			1
Closure			1
Billing (Fees and Interest rates)			1
Account			1
Creation			1
Modification			1
Closure			1
Billing (Fees and Interest rates)			1

From a business perspective, you can investigate "Fees and Interest Rates" on both levels:

- Customer level: all "Fees and Interest Rates" of a Customer
- Account level: "Fees and Interest Rates" of an Account

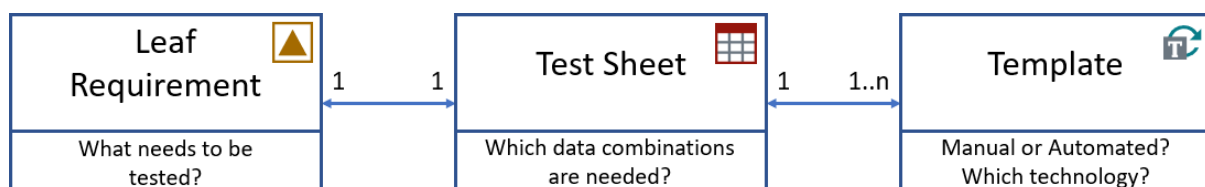
How do you find the correct position? Take a step back and ask yourself: is there a significant difference between the two billing processes? Is there a different risk in the two billing processes?

The solution might also be more specific and could look like this:

Name	Frequ...	Dama...	Weight
Consumer and Business Banking			
Customer			1
Creation			1
Modification			1
Closure			1
Billing (Fees)			1
Account			1
Creation			1
Modification			1
Closure			1
Billing			1
Fees			1
Interest rates			1

4.2.4 Relationship between Requirement, TestSheet and TestCase

For most cases, this is the recommended relationship between Requirement, TestSheet and TestCase Template:

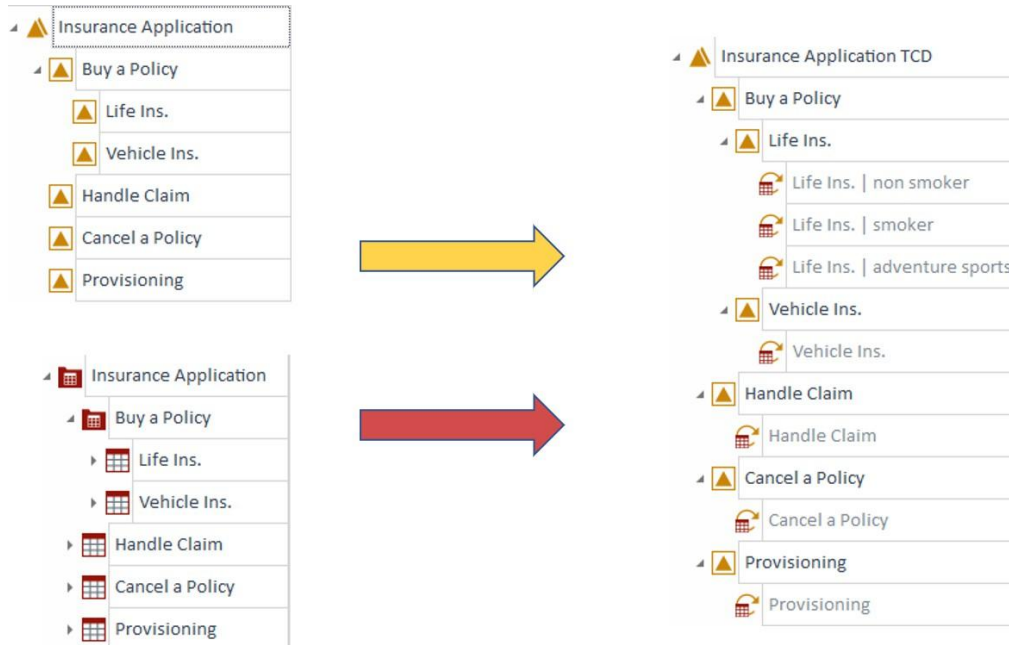


In most cases, one TestSheet can specify the data combinations of one Leaf Requirement. If one TestSheet specifies many requirements (maybe even on one level) than this is an indication that the requirements are too detailed.

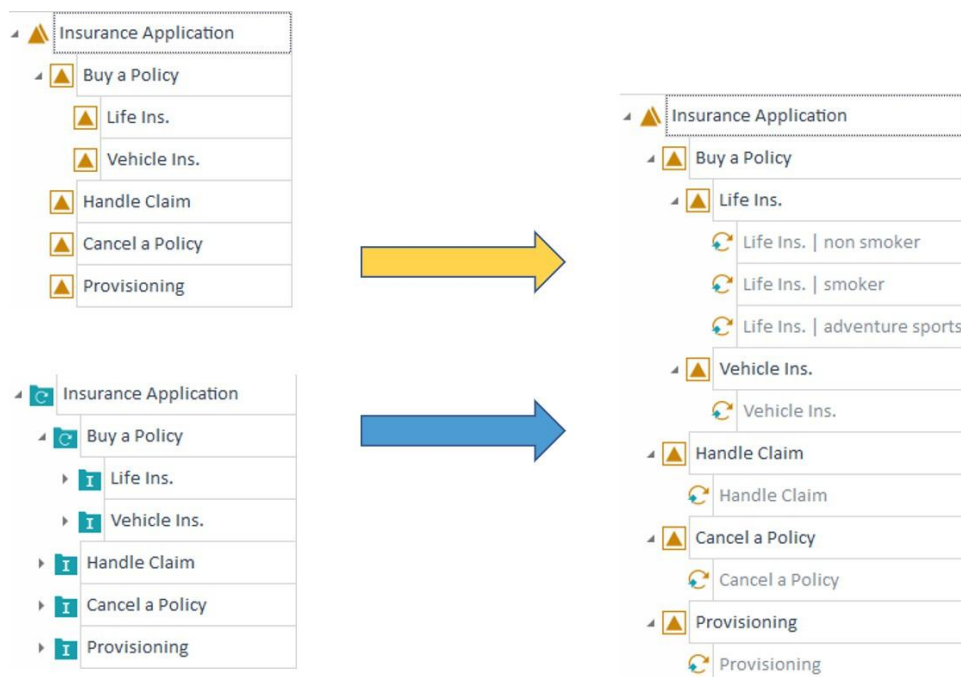
Often one TestCase Template can automate a TestSheet, but in some cases, where the processing is very different depending on the data combination, multiple TestCase Templates are created and linked.

Working with Requirements and TestCases only

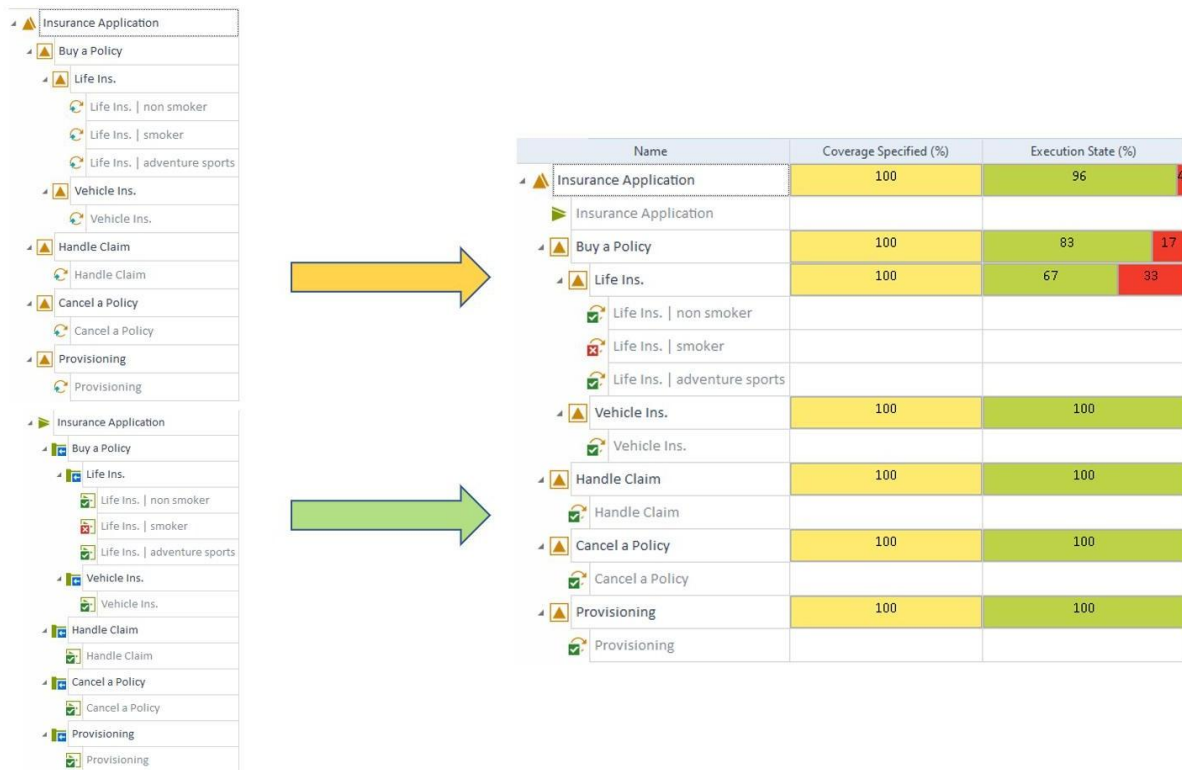
The common approach to start working with Tosca is to build a basic Requirement structure. For each leaf requirement, build a TestCaseDesign and a Template and link them to Requirements.



Usually, one or more TestCases would be linked to one leaf Requirement. In rare cases, a TestCase can be linked to more than one requirement.



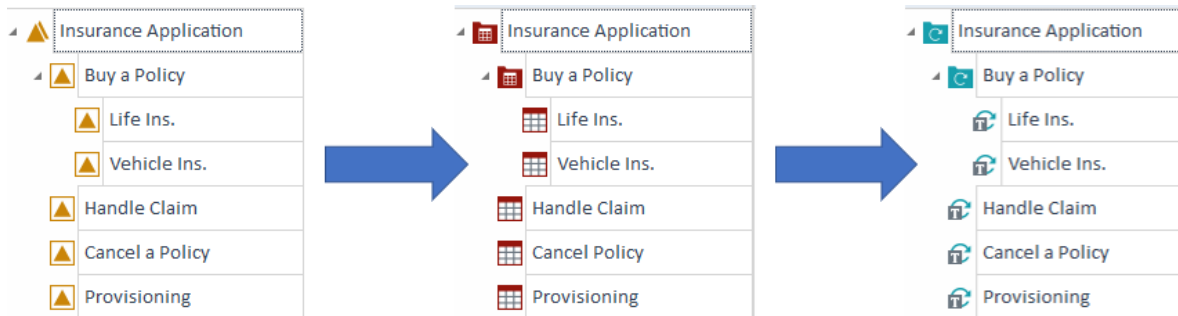
The next step is to add TestCases to ExecutionLists and then link the ExecutionLists back to the Requirements.



Structuring the portfolio

It is important, that the structure created in the Requirement Set is also used in the TestCaseDesign and TestCase sections. When using the same structure, it is easy to link the TestCases to the Requirements and to maintain the structure in the different sections. Another improvement is the navigation within the project, which will be equal in every section.

The next picture shows an example of such a structure.



Hint: There is an easy way to transfer the structure of the requirements section to the other sections:

- Expand Requirement Set
- Mark all elements
- On Requirement Set: Task “Copy table to clipboard”
- On red/blue folder: Task “Create element structure”

4.2.5 Weigh Requirements

Tricentis Tosca offers you tools to calculate the risk, which serves two purposes:

- 1) High Level view of the status of your System.
- 2) Prioritization, where you want to start with automation. To

start with the risk assessment, activate the following columns:

- Weight
- Frequency Class
- Damage Class

Close the Sub-Requirements and focus on the highest level:

	Name	Frequency class	Damage class	Weight
▲ ▲	Example			
▲	Customer			1
▶ ▲	Product			1
▲	Billing			1

- ➔ Weight has a default value of “1” and is calculated based on Frequency Class and Damage Class (see details in Requirements Documentation).

Concentrate on requirements of the same level and choose a value between 1 (very low) and 5 (very high). Consider the following questions:

- Frequency: How often is the functionality used in production? (1 seldom used | 5 often used)
- Damage: What damage happens when the functionality is not working correctly or at all within the production environment (1 not much damage, easy workaround | 5 a lot of damage, no workaround)

Once you have assigned a weight, check whether the values are plausible by considering the following:

- ➔ "Is Requirement A really used more often than B?"
- ➔ "According to the weight, we would focus our tests on Requirement A, right?"

Try to get a big spread between requirements on the same level. When you get a first result of "3" on each position, start analyzing again: Is there absolutely no difference between the requirements concerning usage and damage?

4.2.6 Relative Weight and Contribution

While the weight of a risk is being captured as an absolute number, we also want to know about how a requirement is weighed in relation to other requirements on the same level and in the overall view. The two columns "Relative Weight (%)" and "Contribution (%)" are being calculated for that purpose.

Name	Frequency class	Damage class	Weight	Relative Weight (%)	Contribution (%)
Example					
Customer	3	2	32	10	10
Product	2	3	32	10	10
Product A	3	3	64	80	8
Product B	2	2	16	20	2
Billing	4	4	256	80	80

The relative weight is the weight relative to requirements on the same level.

The contribution represents the weight of an individual requirement compared to the total risk.

Looking at Product A and Product B in the table above, the relative weight on the same level is 80:20 percent whereas the contribution to the overall risk is 8 and 2 percent.

The final view displays all information in detail dependent on the weight:

Name	Frequency class	Damage class	Weight	Relative Weight (%)	Contribution (%)	Coverage Specified (%)	Execution State (%)
Example						100	92
Customer	3	2	32	10	10	100	100
Product	2	3	32	10	10	100	20
Product A	3	3	64	80	8	100	100
Product B	2	2	16	20	2	100	100
Billing	4	4	256	80	80	100	100

5 TestCaseDesign Section (Red)

5.1 Introduction to TestCaseDesign

While the TestCase Section is about doing it right (efficiency), TestCaseDesign is all about doing the right thing (effectivity). Tosca's TestCaseDesign Section supports the testers to efficiently create a TestCaseDesign using TestSheets which can be used to communicate with Subject Matter Experts (SMEs) in business-language. Instead of jumping into TestCase Automation, it is preferred to start with the TestCaseDesign.

The Tosca TestCaseDesign Section enables you to organize TestCases in a logical structure, regardless of whether the test object is available or not. TestCases can then be created with the specification and without the need of an application. A TestCase Template is therefore created in the form of a TestCaseDesign TestSheet as part of the software testing project. This TestSheet allows you to create all combinations of possible TestCases that are required to ensure full test coverage. Equivalence classes serve as a starting point for reducing the number of TestCases needed. The final TestSheet is assigned to the TestCase Template so that all required TestCases are instantiated for execution.

5.1.1 Advantages when using TestCaseDesign

Starting with TestCaseDesign will ensure that the TestCases created during automation will test the right things and that the time spent on creating TestCases is spent as effectively as possible.

TestCaseDesign is about verifying the functionality of the Requirements as effectively as possible. This means focusing on the right data variations and the right verifications to test the Requirements with as few TestCases as possible.

TestSheets in Tosca enable the testers to create a TestCaseDesign with subject matter experts. TestSheets provide the functionality to easily create instances reflecting the different data variations necessary to test a Requirement. TestCase Instances can be generated based on these instances and a TestCase Template.

When looking at TestCaseDesign for automated TestCases, there are different characteristics a high quality TestCaseDesign should have.

5.1.2 General characteristics for good TestCaseDesign

General TestCaseDesign

- TestSheets are based on Requirements and reflect the data variations that need to be tested for the Requirement.
- The Attributes in the TestSheet are described in business language (instead of the more technical language of Modules and TestCases).
- Each TestCase or TestCase Instance is meaningful and understandable – it tests a business functionality and is linked to a Requirement.
- Each TestCase has a clear and defined verification.
- TestCase variations are created based on equivalence classes to provide maximum coverage with minimum redundancy.

Test Data Management

- A TestCase uses stable Test Data so it can be used multiple times.

- No manual steps needed to create Test Data prior running a TestCase.
- If a TestCase consumes Data from another TestCase, take required data from Test Data Service (TDS). The same is valid for TestCases that produce Data for future TestCases.

TestCaseDesign vs. Test Data Service (TDS)

- **TestCaseDesign** should contain data which can be reused for multiple executions. That data should not change over time or between different environments.
 - System Data (e.g.: Warehouse Number, Site Number)
 - Master Data (e.g.: Material Group, Condition Type)
 - Attributes for object types (e.g., specific characteristics of an account)
- **TestCaseDesign** uses data that is known before running the test case
- **TestData Service** contains test data, that usually is not known before test execution, will be created while running the test case or can only be used once
 - Test data preparation
 - Transactional data
 - Consumption data
- In more complex environments, **Test Data Service** can also be used to store information like Logins & Passwords for various system to allow central maintenance

TestCase Execution

- Each TestCase is executable without manual preparations.
- Each TestCase is executable multiple times without manual preparation or cleanup.
- TestCases are grouped into ExecutionLists and each list has a dedicated focus.

Naming and folder structure

- The structure of the Requirement area, the TestCaseDesign section and the TestCase section should ideally match.
- TestSheets and Attributes have meaningful, business-related names.
- Ideally, there should be 6-8 Attributes per level.
- Build a meaningful structure in business language!
- Do include business relevant input / output information.
- Do not replicate the complete test case flow in all technical detail.

5.1.3 The TestSheet

TestSheets are the basic framework in TestCaseDesign. They map the TestCase structure and the included values.

You can build data oriented TestCases by using TestCaseDesign Attributes.

Concrete TestCase values are referred to as TestCaseDesign Instances, i.e. the columns shown in the illustration below. The Attribute Values are assigned to these Instances.

Additional information about the TestSheet can be also found in the [Tosca online manual](#).

5.1.4 When to use TestCaseDesign – and when not to

Most times using TestCaseDesign is the preferred way to work with Tosca. Only when using TestCaseDesign can you leverage the full feature set of Tosca including Combinatorics,

Equivalence class definition, Template use and Instance creation.

Still, there might be reasons against creating a TestSheet – as listed below:

- The necessary TestCases (and Instances) are well known and only need to be automated. In this case, it would often still make sense to use TestCaseDesign, but it is also possible to create “plain” TestCases.
- Your TestCase is going to be used only once (only focused on actual sprint)
- The TestCase serves the purpose of kicking off a technical job (e.g. Batch) – which is static and always works the same way.
- The Business functionality which should be tested doesn’t have multiple paths or input variations and thus needs only one TestCase for each workflow – which is very seldom. In this case, a TestCaseDesign often still makes sense to extract the TestData from the business process.

5.1.5 How to get started

There are different ways to create a new TestSheet. The most common way is to create a new TestSheet from the scratch. If there is already an existing and fitting TestCaseDesign description in Excel, the TestSheet can be created based on the Excel content as well.

Create a TestSheet from scratch

 [Create a new TestSheet](#)

Create a TestSheet with Excel Input

Excel content can be used to create a TestSheet if it is formatted the correct way. Here is a screenshot that shows the necessary format in the Excel Sheet.

	A	B	C	D	E
1			Debit Security <=1000	Credit >1000	Credit Security <=1000
2	Precondition				
3		Security	Yes	No	Yes
4		Client	John Doe	Joanna Doe	Jack Johnson
5	Process				
6		Transaction Type	Debit	Credit	Credit
7		Quantity	2	1	3
8		Price	1000	3000	500
9	Verification				
10		Pending Quantity	1	0	1
11		Executed Quantity	1	1	2

After creating a new TestSheet, use right-click to open the context menu and click on “Create Element Structure from Clipboard” in “...” to create the TestSheet content.

A Template can be found in following location:

%Tricentis_Projects%\ToscaCommander\Templates\BO\

The TestSheet should then look like this:

Name	<input checked="" type="checkbox"/> Debit Security <= 1000	<input checked="" type="checkbox"/> Credit > 1000	<input checked="" type="checkbox"/> Credit Security <= 1000
Transactions Security	Debit Security <=1000	Credit >1000	Credit Security <=1000
Instances			
Precondition			
Security	Yes	No	Yes
Client	John Doe	Joanna Doe	Jack Johnson
Process			
Transaction Type	Debit	Credit	Credit
Quantity	2	1	3
Price	1000	3000	500
Verification			
Pending Quantity	1	0	1
Executed Quantity	1	1	2

Limitations of this approach:

- Due to the missing information when inserting data from Excel, no instances are created and no information about Straight Through Instances or other instance related information gets imported.
- This can be especially painful when exporting TestSheets to Excel, modifying them and importing them again. This is sometimes an approach preferred by SMEs if they are used to working with Excel.

The other way around is also possible. Collapse everything you want to copy within the TestSheet, mark everything (CTRL + A) and copy it with the structure (CTRL+SHIFT+C). Paste as usual within an Excel map.

5.2 TestSheet - Best Practices

5.2.1 Basic TestSheet Structure

To use TestSheets efficiently, a certain high-level structure should be implemented:

Name	<input checked="" type="checkbox"/> Standard Order	<input checked="" type="checkbox"/> Rush Order	<input checked="" type="checkbox"/> Sales Organizat...	<input checked="" type="checkbox"/> External	<input checked="" type="checkbox"/> Distribution	<input checked="" type="checkbox"/> Sales Office 1	<input checked="" type="checkbox"/> Sales Office 2	<input checked="" type="checkbox"/> Sales Office 3	<input checked="" type="checkbox"/> Sales Office 4
Sales Order prepared (S/4)	Standard Order	Rush Order	Sales Organizati...	External	Distribution	Sales Office 1	Sales Office 2	Sales Office 3	Sales Office 4
Instances									
Administration									
TC Designer	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe	<input checked="" type="checkbox"/> John Doe
SME	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe	<input checked="" type="checkbox"/> Jane Doe
Precondition									
System ID	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...	<input checked="" type="checkbox"/> SAP IDES Int...
SAP Logon/Login	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES	<input checked="" type="checkbox"/> SAP IDES
Process									
Order types	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Rush Order	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Standard Or...	<input checked="" type="checkbox"/> Standard Or...
Sales Organizations	<input checked="" type="checkbox"/> Spectar Mel...	<input checked="" type="checkbox"/> Spectar Mel...	<input checked="" type="checkbox"/> 3010	<input checked="" type="checkbox"/> TRC	<input checked="" type="checkbox"/> 01	<input checked="" type="checkbox"/> Sales Office 1	<input checked="" type="checkbox"/> Sales Office 2	<input checked="" type="checkbox"/> Sales Office 3	<input checked="" type="checkbox"/> Sales Office 4
Customers	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar	<input checked="" type="checkbox"/> Spectar
Materials	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...	<input checked="" type="checkbox"/> Standard Ma...
Value	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000
Verification									
Invoice Amount	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000	<input checked="" type="checkbox"/> 1000

- Administration | Business Relevant = No
 - TC Designer: Information about the person that created the TestSheet
 - SME (Subject Matter Expert): Business Expert that has the knowledge about the process
- Precondition | Business Relevant = No
 - Contains all information that is required to run the test case, e.g.:
 - Business Preconditions
 - System Information
- Process | Business Relevant = Yes
 - Contains all business information in a structured form fitting to the process flow
 - There should be a maximum of 8-10 Attributes on one level
- Verification | Business Relevant = Result
 - Contains the verification point(s) within the test case

5.2.2 Generating Instances using Combinatorics

- [How to Create Instances Using Combinatorics](#)

Combinatorics are a great way to achieve the ideal amount of test cases. However, in case there are lots of dependencies between the different test data used in a test case, the benefit of using combinatorics is limited.

5.2.3 Using equivalence classes and specific values

Equivalence classes are ranges of values that can be considered equal in a test case. For example, if you have an application that applies a standard to all people aged 20 to 60 years old, then any value in that range can be considered equivalent and only one value in the range needs to be tested. If the test passes for someone aged 21, then it can be assumed it will pass for someone aged 45 since they are in the same equivalence class.

When testing new functionality, you also need to test specific values to validate the boundaries. In the previous example, boundary values would be 19 and 61, and instances of the test case would need to be created to test those values to ensure that the standard is not applied to these instances.

Specific boundary value instances are not needed in a regression test, as functionality around the boundaries is not likely to change from release to release.

- [Click here for a TestCaseDesign example](#) that uses both equivalence and specific boundary examples.

5.2.4 Linking TestCases with TestCaseDesign

To use all the specified data from TestCaseDesign in the TestCase section, a Template should be created and linked to a TestCaseDesign Sheet. That easily happens via drag & drop.

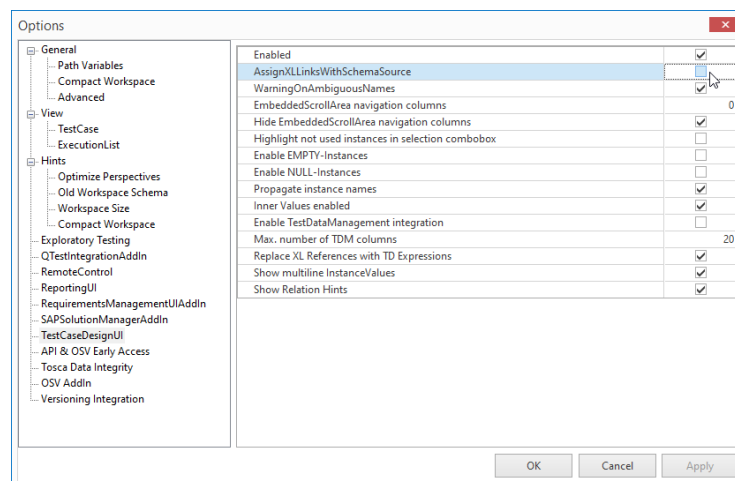


Name	1	2	3	4	5	6	7	8	9	10
Instance Calculator New	StraightThrough	Gold	Platinum	Ultimate	No Starting Date	3.000.000,00	5.000.000,00	10.000.000,00		
Instances										
Administration										
Precondition										
Process	StraightThro...	Gold	Platinum	Ultimate	No Starting D...	3.000.000,00	5.000.000,00	10.000.000,00		
Instances										
Vehicle	Private	Private	Private	Private	Private	Private	Private	Private	Private	Private
Insurant	Other	Other	Other	Other	Other	Other	Other	Other	Other	Other
Product	StraightThro...	StraightThro...	StraightThro...	StraightThro...	No Starting D...	3.000.000,00	5.000.000,00	10.000.000,00		
Price Option	Silver	Gold	Platinum	Ultimate	Silver	Silver	Silver	Silver	Silver	Silver
Error Message	No	No	No	No	No	No	No	No	No	No
Verification										
Yearly Amount [USD]										

IC_Automobile
TemplateInstance of IC_Automobile
StraightThrough
Gold
Platinum
Ultimate
No Starting Date
3.000.000,00
5.000.000,00
10.000.000,00
Euro Protection
Legal Defense Insurance

Within the Options Dialogue, the “TestCaseDesignUI” Option “AssignXLLinksWithSchemaSource” is responsible for creating automatic links between XTestStepValue Names and TestCaseDesign Module Attribute Names during the drag & drop operation. This functionality can lead to confusion and needs be used carefully.

- Beneficial when in the early steps of TestCase creation
- Should be deactivated when working with existing TestCases in more complex portfolios



5.2.5 How to reduce maintenance effort further: Classes

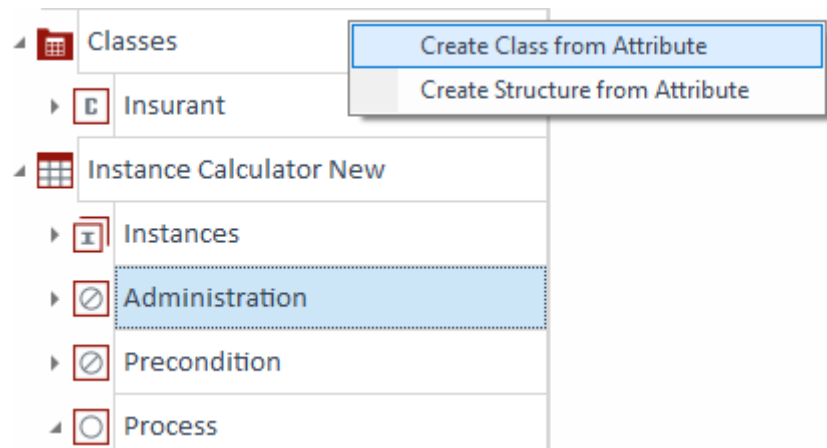
TestCaseDesign Classes are ideal for attributes, which are repeatedly used. These classes are centrally managed and used at several positions as class references.


5.2.5.1 How to create Classes

When creating a TestSheet, you might notice that you are going to use values which will be reused frequently, e.g. addresses or ranges of numbers. As a rule of thumb, whenever you intend to copy a certain Attribute or structure from an existing TestSheet, it might be worth creating a class for it.

You can handle this easily:

1. Select the attribute(s) to be saved as a class.
2. Use the context menu or drag & drop to move the attribute(s) to the TestCaseDesign folder that should contain the new TestCaseDesign Class.
 - ➔ Tricentis Tosca™ now creates a new TestCaseDesign Class with the name of the attribute. The attribute itself is replaced by a reference to this TestCaseDesign Class. The name is not modified.



 [How to create Classes.](#)

5.2.5.2 How to use Classes

Classes are used as a reference, which can either be created in another class, in a TestSheet, or an Attribute.

1. Select the TestCaseDesign Class to be used as a reference. You can also select more than one class.
2. Drag your selected objects to the required target object and drop it there. One or more Class References will be created.



6 Modules Section (Orange)

Modules are the building blocks of your TestCases. They contain the technical information that Tricentis Tosca uses to perform a sequence of automated actions on your system under test.

6.1 Structure and Naming

6.1.1 Structure and Naming of Modules

As every customer usually has a complex environment with different applications, the Module name should precisely show what the Module reflects in which application. An example may look as follows:

Vehicle Insurance Calculator (VIA)	
Vehicle Insurance Calculator (VIA)	1
General	2
VIA General Login	
VIA General Logout	
Popups	3
VIA Popup Information	
VIA Popup Remember User	
VIA Popup Quote Confirmation	
Automobile	2
VIA Automobile Enter Vehicle Data	
VIA Automobile Enter Insurant Data	
VIA Automobile Enter Product Data	
VIA Automobile Select Price Option	
VIA Automobile Send Quote	

Demo Web Shop (DWS)	
Demo Web Shop (DWS)	1
General	2
DWS General Login	
DWS General Logout	
Popups	3
DWS Popup Information	
DWS Popup Remember User	
Account	2
DWS Account Overview	
DWS Account Settings	
Product	2
DWS Product Catalog	
DWS Product Shopping Cart	
Payment	2
DWS Payment Billing Data	
DWS Payment Payment Data	
DWS Payment Confirmation	

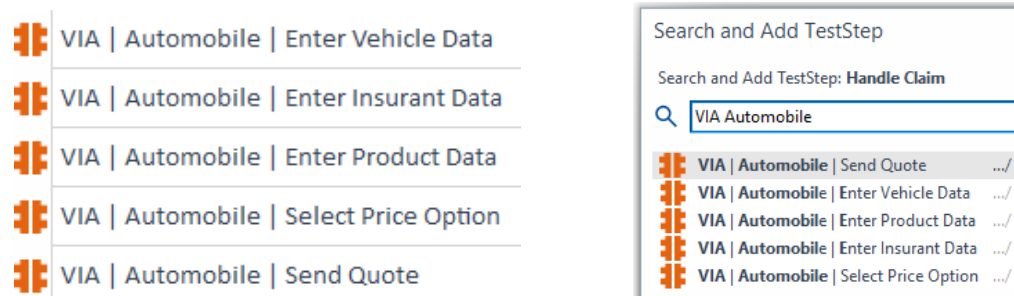
- 1 Application Folder Name
- Business readable application name
 - Short abbreviation of application (e.g.: VIA / DWS)

- 2 Function/Area Name
- Business readable name
 - Named according to the area / functionality the modules are covering, e.g.: General
Automobile
ME2xN | Purchase Order (ME21N & ME22N & ME23N → ME2xN) WE01 |
Invoice Matching

- 3 Subfolder Name

- Business readable name
- Named according to containing parts (e.g.: Popups, Header Area, Tabs, ...)
- Module name
 - Represented part of the screen, e.g.
 - Everything within the TAB "Org. Data"
 - Window for Billing Data in the Payments area
 - Login Screen for a certain application

By following a certain Module name structure, it will be easy to find the right Modules when searching for them and by using the functionality "Search and Add TestStep (CTRL + T)":



The naming convention shall therefore be as follows:

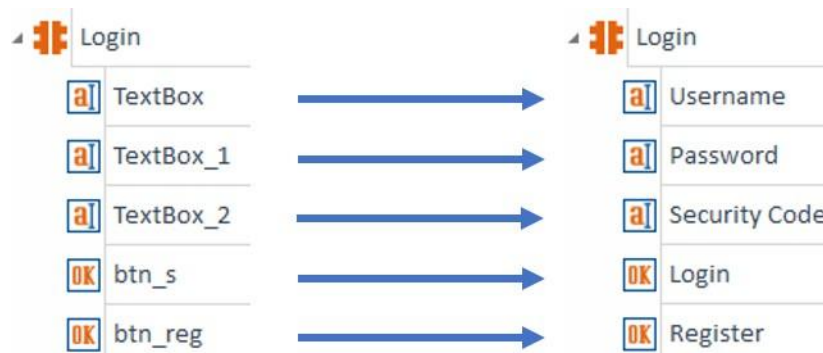
<Application Abbreviation> | <1...n Structure of where the module is in> | <Part of the screen the Module covers>

Examples:

- VIA | Automobile | Send Quote
- DWS | Account | Overview
- SAP | General | Popup | Information
- Fiori | Adjust Stock | Add Article
- ...

6.1.2 Naming of ModuleAttributes

ModuleAttributes should be named after the definition of the field on the screen. Typically, these names are provided by labels next to the control. If that is not the case, a business-related meaning will help to identify the correct control on the Module. In any case, the name should be in a human readable form and not a technical definition.



6.1.3 Breaking down Modules

If a screen contains many fields, the Module should be broken down into additional Modules which define a business-related grouping for those controls. This grants maintainable, readable Modules and adds additional flexibility to them, since not every test case will require every field / section of a screen. It also allows the combination of those Modules in any sequence as needed.

Ideally, this kind of grouping is represented on the application by visual sections. If this is not the case, the fields might be structured by certain requirements to which they belong.

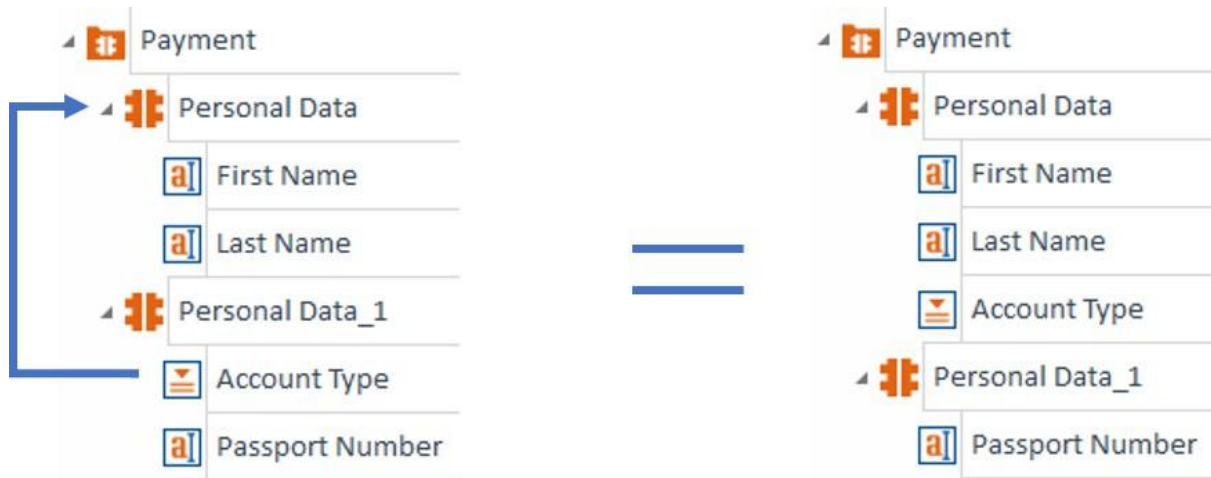
The screenshot shows a complex web form titled 'Add Shopping Instruction'. Red arrows point from various sections of the form to a list of modules on the right. The modules listed are:

- Shopping Instruction
- Asset Details
- Header Information
- Instruction Information
- Shop Status
- Disposition
- Report
- Send Report Button List
- Generic Button List
- Page loading

In general, a Module should not have more than 20 controls. There might be exceptions but going beyond this will lead to additional maintenance effort and a more difficulty in finding required ModuleAttributes.

However, Modules with controls that are only relevant to a specific test case at the time of the scan should be avoided. This very often leads to Modules with only 1-2 controls, and new Modules being needed every time a test case faces new fields on the screen.

Tosca allows you to add additional Attributes to already existing Modules. To accomplish this, simply select a ModuleAttribute from a Module, drag it to another Module, and drop it directly on the Module icon. Your ModuleAttribute will be added to the end of that Module. If you moved all Attributes from one Module to another, the empty Module can simply be deleted. Always consider updating existing Modules.

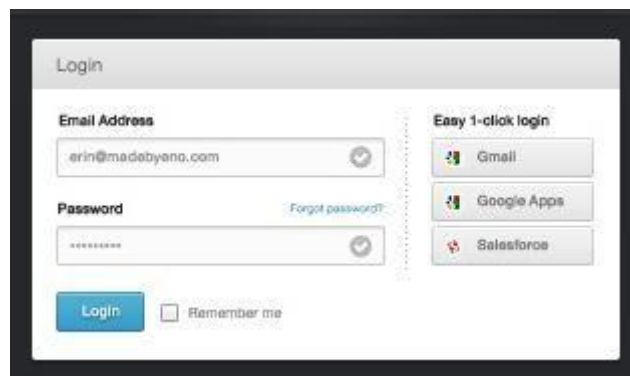


When working with Web Applications, there might be an invisible structure within the Module to consider IFrames and HTML Documents. These can be viewed with the shortcut "F12".

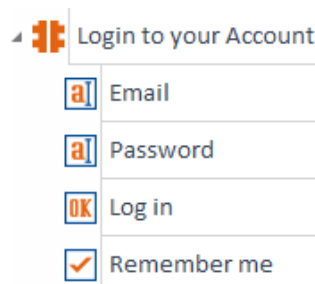
6.1.4 Sequence of ModuleAttributes

During execution, Tosca steers the controls in the sequence they are defined on the module. Since the sequence might not be appropriate, the Attributes can be re-arranged via drag & drop. This is very important and depends on different factors such as the scan not returning a proper order, or a sequence that leads to additional test steps because it does not comply with the business process / flow.

The following login screen is an example. It shows four fields: the email address, the password, a login button and a checkbox to remember the provided login details.



By scanning this with Tosca, the Module will show it in the structure as it appears on the screen, with the Remember Me checkbox being the last attribute.



While this looks pretty, it comes with a downside. If the user wants to select the Remember Me function, it will require at least two test steps.

	Name	Value	ActionMode
↻	Login to your Account		
↻	Enter Login Details		
a	Email	test@tester.com	Input
a	Password	*****	Input
OK	Log in		
✓	Remember me	True	Input
↻	Click on Login		
a	Email		
a	Password		
OK	Log in	X	Input
✓	Remember me		

This might not be ideal, especially for such a small screen. Since there is never a use case where the remember me would follow the Log In button, it makes sense to rearrange the sequence, having the checkbox steered before the Log In button. This rearranging can be achieved with simple drag & drop. A black line will appear indicating where the attribute will drop in-between.



This order represents a much more appropriate process for the screen, accomplishing the same actions in one step:

	Name	Value	ActionMode
↻	Login to your Account		
↻	Login to your Account		
a	Email	test@test.com	Input
a	Password	*****	Input
✓	Remember me	True	Input
OK	Log in	X	Input

The sequence should always be arranged thoughtfully. Modules must stay reusable since the same field should not occur in more than one Module. Therefore, it is not recommended to

change the arrangement just because of a few test cases if most the test cases are affected.

6.1.5 Duplicate controls and cardinality

Every field on the screen should only be covered once in the Module. Duplicated ModuleAttributes (representing the same control) should be avoided. If multiple operations are necessary, they can be split over multiple test steps. Below is a simple example.

	Name	Value	ActionMode
▶	Submit Transaction		
▶	Wait and Search		
▶	Search	Exists == True	WaitOn
▶	Search	X	Input

VS.

	Name	Value	ActionMode
▶	Submit Transaction		
▶	Wait for Search		
▶	Search	Exists == True	WaitOn
▶	Search		
▶	Search	X	Input

Duplicates lead to additional maintenance effort, e.g. if the identification of the control changed. In addition, they can drastically increase the size of a module, impacting readability and causing confusion.

This also includes the use of cardinality for GUI modules. There are some situations where cardinality might come in handy, but it is rarely used for GUI. Please see further details in the [online manual](#).

6.2 Creating Modules correctly

6.2.1 Scanning and Steering

A technical ID in Tosca identifies each object that is used for an automated test case. Technical IDs are control properties like "Name", "InnerText" and so on. The property that is selected for identification must be unique. Sometimes a single property is not enough, but maybe two or three combined make the control steerable without using other options. Use Wildcard (*) to ignore parts of the ID that might not be unique.

Example of good identification:

Name	Type	Value
InnerText	XParam	Continue
DefaultName	XParam	Cost Center
AccTooltip	XParam	Search Criteria:*
attributes title	XParam	Display Repair
Caption	XParam	Invoice*
DefaultTooltip	XParam	Delivery
Enabled	XParam	True
IconName	XParam	B CANC
Id	XParam	downloadReferesh
Label	XParam	status-in-process
LocalName	XParam	term
Name	XParam	/SAPAPO/MATIO-MATNR
RelativeId	XParam	/usr/ctxtRC29N-MATNR

Example of bad identification:

Name	Type	Value
InnerText	XParam	
Title	XParam	
InnerText	XParam	Notification Number Read BarCode
ActionPoint	XParam	"{X=544.5, Y=628}"
Caption	XParam	USD 0.00
ControlArea	XParam	"{X=136,Y=459,Width=76,Height=20}"
DefaultName	XParam	txtPersonas 154661101511135 *
DefaultName	XParam	button372
Id	XParam	fancybox-frame1527697275982
OuterHtml	XParam	"<input id=""WD61"" ct=""CB"" Isdata=""{2:'WD62',3:'MY CR PART',4:'To\x20Be\x20..."
Title	XParam	Organization: 40137872, AutoAP HK FRIEND & FAMILY SALES / Hong Kong - Sales Ar...
Text	XParam	"<html><p width=""188"" style=""text-align: left;""> Suite</p></html>"

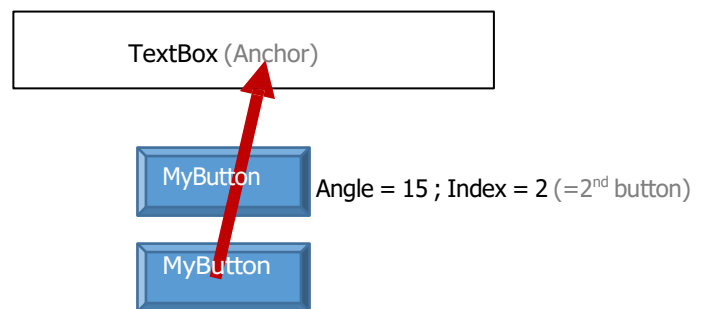
If no unique property / combination of properties can be found to identify a control uniquely, there are several options to identify such controls. These options are described below.

6.2.1.1 Anchor objects

Controls that are used as anchor objects to identify other objects should have stable IDs given by the developers.

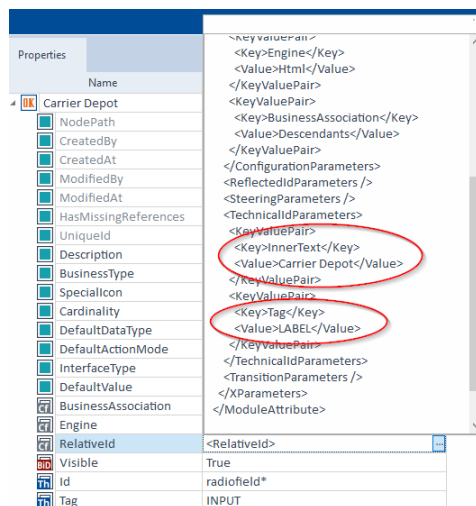
You will find more information about identification via anchor objects in the online documentation.

Location-based verification is based on using anchor objects. When scanning a Module, you can use an anchor object as an identifier. Tosca will then identify the angle and the number of objects of the same type on that vector.



Finding the anchor object:

- The anchor object is stored in the RelativeID parameter of the module. There, the entire information is stored:



- In the TechnicalIdParameters Node, the anchor object is being identified and you can see which object is taken as anchor.

Important when using the anchor for location-based verifications:

- Ensure that "RelativeId" is the only TechnicalID in the module. If you provide more than only the "RelativeId" there will be more IDs of the object found independent of the location.
- [Link to the online documentation](#)

6.2.1.2 Prevent Anchors by Using Parent Objects

When doing a new scan always check if the parent node has a unique ID, then select it and select the object below that was previously not unique. In most of the cases you can prevent other methods like the use of anchors or image based identification.

The screenshot shows the Tricentis XScan interface with the 'Identify by Properties' pane open. The tree view on the left shows a hierarchy of objects. The 'Vehicle Insurance Application This is a sample application, Version 1.0.1' object is selected, and its 'DIV' child is highlighted. The 'Identify by Properties' pane shows the following properties:

Representation	
ActionPoint	"{X=1072,5, Y=124...
Adapter	Tricentis.Automation...
AssociatedLabel	<No label associated>
Context	Tricentis.Automation...
ControlArea	"{X=928,Y=100,Wid...
DefaultName	Vehicle Insurance A...
Enabled	True
Focused	False
Id	
InteractiveElement	False
IsSteerable	True
Technical	Tricentis.Automation...
Text	Vehicle Insurance A...
Visible	True
VisualSelectionPriority	Default

The 'Technical' section shows:

Technical	
ClassName	logo-text
Id	
InnerHTML	*
InnerText	

At the bottom, a status bar indicates: '1 item is not unique' and 'The selected item is not unique!'.

The screenshot shows the Tricentis XScan interface with the 'Identify by Properties' pane open. The tree view on the left shows the same hierarchy. The 'Vehicle Insurance Application This is a sample application, Version 1.0.1' object is selected, and its 'DIV' child is highlighted. The 'Identify by Properties' pane shows the following properties:

Representation	
ActionPoint	"{X=959,5, Y=124,...
Adapter	Tricentis.Automation...
AssociatedLabel	<No label associated>
Context	Tricentis.Automation...
ControlArea	"{X=702,Y=100,Wid...
DefaultName	
Enabled	True
Focused	False
Id	branding
InteractiveElement	True
IsSteerable	True
Label	
Technical	Tricentis.Automation...
Url	http://sampleapp.tri...
Visible	True
VisualSelectionPriority	Default

The 'Technical' section shows:

Technical	
ClassName	
Href	http://sampleapp.tri...
Id	branding

At the bottom, a status bar indicates: 'The selected item is unique.'

Important when using parent controls for identification:

- In case Popups are loaded as part of a screen, it is good practice to include the popup itself within the scanned Module. The top-level container that reflects the page can usually be very easily uniquely identified by a property like "ClassName" or "InnerText".
- In many cases, the correct parent that you need might not be the one directly above your control. You can find the appropriate parent when clicking through the parents having the option "Highlight Selection" activated in the XScan.

6.2.1.3 Image based identification

Controls could be identified by an image of the control itself or an anchor image. However, if the resolution or the design of your application changes, you might run into a control identification error. Multiple images can be added for one control to ensure increased stability.

You will find more information about identification via anchor objects in the online documentation.

- [Link to the online documentation](#)

6.2.1.4 Identify by Index

Controls, like links or buttons which could appear multiple times within an application, could be identified by using of an internal Tosca Index. However, keep in mind, that if an additional control with the same technical information will be added to your application, you might run into a control identification error.

To steer a specific control, you don't always need to select an index within the XScan. It is possible to use dynamic numbers, buffers or links to a TestSheet to specify an index which should be used during execution.

You can find more information in the online documentation.

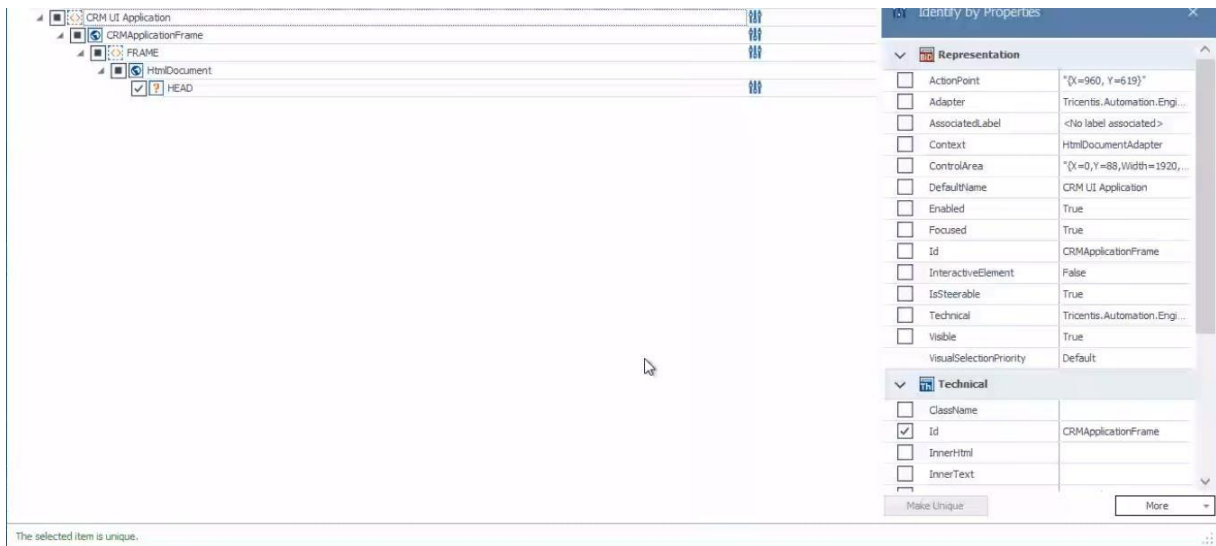
- [Link to the online documentation](#)

6.2.1.5 Location-based objects – Naming

All controls which were scanned for location-based verifications have the ending "_LocBased" to be sure that this verification is for getting a location. Therefore, it is necessary that this control is identified by anchor only (InnerText or further properties must not be used).

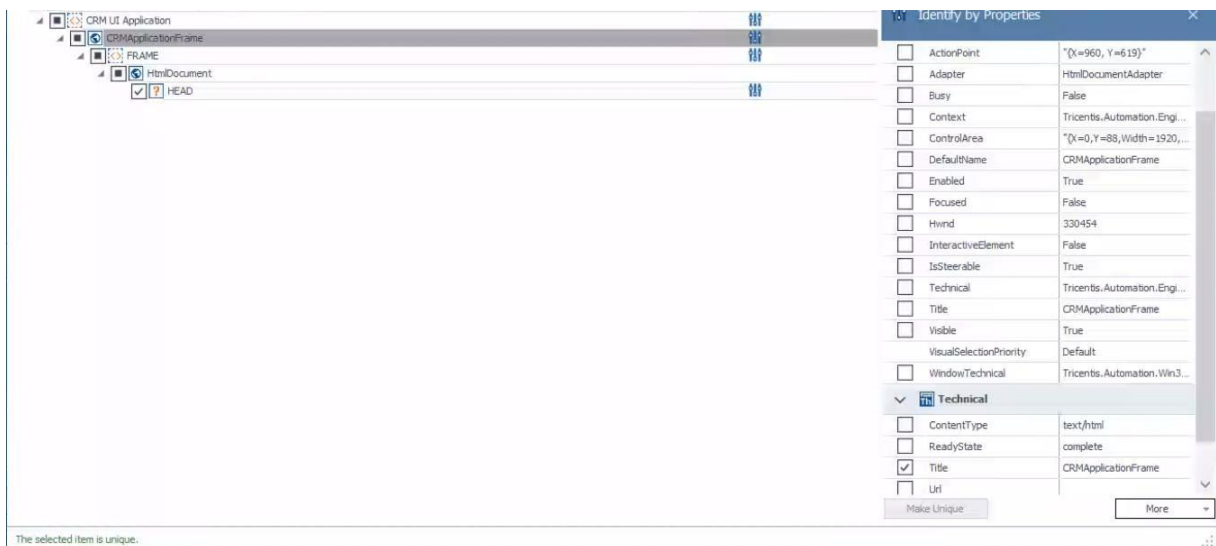
6.2.1.6 Pages with IFrames

When scanning pages, it is common that IFrames appear with auto generated IDs (or identifiers) which can change over the lifecycle of the application. An example can be seen in the screenshot below.



In the screenshot above, the generic element is in an IFrame. There are two sources of instability:

- 1) IFrame ID
- 2) Title of the HTML Document



If there is certain text in the Title that only belongs to a specific test case you can use a wildcard instead of a specific number (e.g. "Case:*" instead of "Case: 02956254...").


Within the Commander you can see the IFrames, if hidden, by clicking F12 in the modules section.

6.2.1.7 Hints on Control Identification & Steering

As a reminder on topics that were discussed during Training and Hands-On coaching:

- Be aware of the Caption on Module level
 - Example: The caption of the Browser window after entering an order says "Order: 11456"
 - The controls will be the same **for every order**

- Therefore, the caption should be changed to "Order*" (Wildcard) so the Module can be reused, independent of the order number
- Be aware of ID's – just because the Scan says the control identification is Unique, this doesn't mean it will stay unique
 - Find a Unique AND Stable combination to identify controls
 - Use Properties, Anchor controls, Image based identification or Index

- 
 - Be aware of ID's – they are probably dynamically generated and could be not stable after reopening the application or if users with other responsibilities have more rights and therefore see more controls.

6.3 Standard Modules

6.3.1 Purpose

Tosca provides certain Standard Modules. They look the same as Modules that have been scanned, with the difference that they perform special functions.

Those can be e.g., sending certain keys, launching applications, or closing windows. There are engine-specific Modules and general Modules, which provide common functionality.

The "standard.tsu" contains all required TBox Modules. That includes common functionality such as send keys, but also provides engine-specific functionality.

Special functions: [Online manual link](#)

TBox Automation Tools: [Online manual link](#)

6.3.2 When to use which Standard Module

The rule should be to use a Standard Module related to the framework that the main TestCase reflects (OpenURL for a test case which reflects e.g. XBrowser).

6.4 Maintaining Modules

6.4.1 Using Rescan

The Rescan functionality allows you to add or modify controls on your XModule. It is like the scan behavior with the difference that it will populate the controls that you have already selected on the screen.

This is commonly useful when there is an update on the application and identification needs to be adapted, or when a control wasn't selected in the original scan.

If the rescan cannot match a control by its identification criteria, then it will be shown in a list of unknown controls, allowing you to reassign it to a control on the screen.

The rescan option might not always be available. In that case, there are other ways to maintain the XModule. Further Information can be found in the online manual:

- [Link to the online documentation](#)

6.4.2 Replacing Modules

Another way to update a Module is to replace it with a new Module. This is possible by pressing the shift key and dragging the old Module onto the new Module.

This way, value ranges, identifications, and additional controls can be updated in all the existing test steps.

This option will not be provided if there is no test step referring to the source Module, since it can simply be deleted.

Please find details on the usage in the [manual](#).

6.4.3 Modifying Module information

Adjustments of specific controls can be done on the Module directly without a rescan / replacement.

When a ModuleAttribute is selected, the properties tab will show all the identification and steering properties that are defined to identify and use the control on the screen.

As example, a control might be identified by an Id. It is possible to adapt this Id if it turns out to be dynamic, using e.g. a regular expression or a wildcard (*).

Properties		
	Name	Value
✖	Email	
🔍	NodePath	/Best Practices/Modules/Module Inform...
🔍	CreatedBy	Admin
🔍	CreatedAt	14.04.2023 18:16:25
🔍	ModifiedBy	Unknown
🔍	ModifiedAt	Unknown
🔍	HasMissingReferences	False
🔍	Uniqueid	3a0a92ba-9747-06d0-31e3-b2994d2d2407
🔍	Description	
🔍	BusinessType	TextBox
🔍	SpecialIcon	
🔍	Cardinality	0-1
🔍	DefaultDataType	String
🔍	DefaultActionMode	Input
🔍	InterfaceType	Implicit
🔍	DefaultValue	
🔍	BusinessAssociation	Descendants
🔍	Engine	Html
🔍	FireEvent	change
🔍	Id	DynamicId_*
🔍	Tag	INPUT

Or:

Properties		
	Name	Value
✖	Email	
🔍	NodePath	/Best Practices/Modules/Module Inform...
🔍	CreatedBy	Admin
🔍	CreatedAt	14.04.2023 18:19:27
🔍	ModifiedBy	Unknown
🔍	ModifiedAt	Unknown
🔍	HasMissingReferences	False
🔍	Uniqueid	3a0a92bd-5cb9-a131-5ee9-96b2d811cb26
🔍	Description	
🔍	BusinessType	TextBox
🔍	SpecialIcon	
🔍	Cardinality	0-1
🔍	DefaultDataType	String
🔍	DefaultActionMode	Input
🔍	InterfaceType	Implicit
🔍	DefaultValue	
🔍	BusinessAssociation	Descendants
🔍	Engine	Html
🔍	FireEvent	change
🔍	Id	{REGEX["Email"]}
🔍	Tag	INPUT

6.4.4 Scrolling Behavior

How to change the scrolling behavior in Tosca:

Add the following Steering Parameter to a ModuleAttribute. Instead of scrolling to the center (default) the definition of "bottom" or "top" can be modified here.

In the module section, attributes like in the screenshot below and a new steering parameter can be created.



Properties	
Name	Value
Email	
NodePath	/Best Practices/Modules/Module Inform...
CreatedBy	Admin
CreatedAt	14.04.2023 18:20:32
ModifiedBy	Admin
ModifiedAt	14.04.2023 18:22:02
HasMissingReferences	False
Uniquelid	3a0a92be-59fe-203c-6213-f6508f80f53d
Description	
BusinessType	TextBox
SpecialIcon	
Cardinality	0-1
DefaultDataType	String
DefaultActionMode	Input
InterfaceType	Implicit
DefaultValue	
BusinessAssociation	Descendants
Engine	Html
FireEvent	change
ScrollingBehavior	Top
Id	{REGEX["Email"]}
Tag	INPUT

6.4.5 User Simulation

Sometimes you must simulate a user input via Tosca.

This can be done by adding the following SteeringParameter to a ModuleAttribute. Simply create a Steering parameter as shown in the previous chapter and set the value to "True."

Examples where UserSimulation may be needed:

- Responsive Controls (e.g. Text Field with auto complete)
- Buttons that open modal windows
 - Ignore what happens after the click

Properties		
	Name	Value
✖ a	Password	
■	NodePath	/Best Practices/Modules/Module Inform...
■	CreatedBy	Admin
■	CreatedAt	14.04.2023 18:20:32
■	ModifiedBy	Admin
■	ModifiedAt	14.04.2023 18:24:58
■	HasMissingReferences	False
■	UniquelId	3a0a92be-59ff-9938-e15b-3cad9e14b129
■	Description	
■	BusinessType	TextBox
■	SpecialIcon	
■	Cardinality	0-1
■	DefaultDataType	String
■	DefaultActionMode	
■	InterfaceType	Implicit
■	DefaultValue	
cf	BusinessAssociation	Descendants
cf	Engine	Html
s	FireEvent	change
s	UserSimulation	True
Th	Id	Password
Th	Tag	INPUT

7 TestCase Section (Blue)

7.1 Tips on TestCase Automation

The next chapters contain examples, lessons learned and tips from previous customer projects.

7.1.1 Usage of TestCaseWorkState

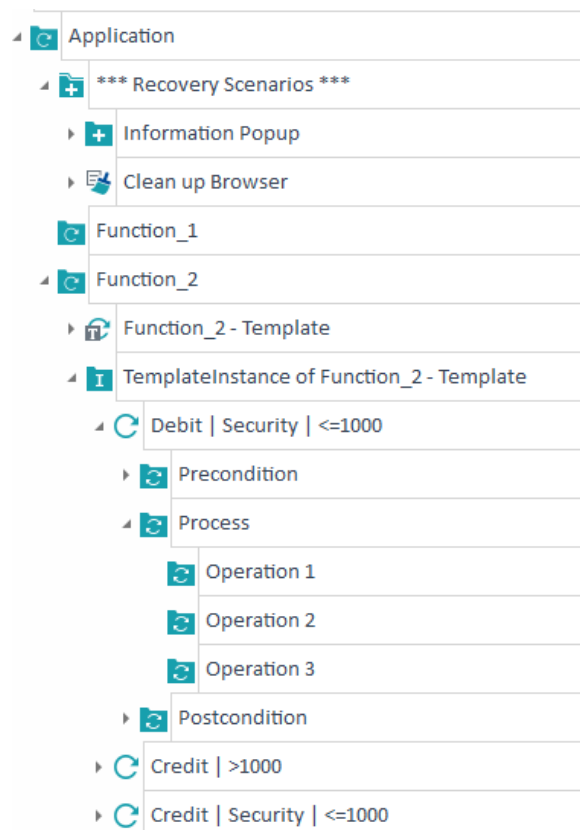
TestCases, by default, have a property called TestCaseWorkState which describes the working state of the TestCase. Available values are

- PLANNED – The TestCase is planned
- IN_WORK – The TestCase is currently in progress
- COMPLETED – The TestCase is completed

Upon creation of a TestCase the TestCaseWorkState is PLANNED. As soon as you start working on a TestCase you should change the TestCaseWorkState to IN_WORK. Once the TestCase is finished, set the value to COMPLETED. This ensures to display the correct coverage in the requirement section.

7.1.2 Standard folder structure within a TestCase

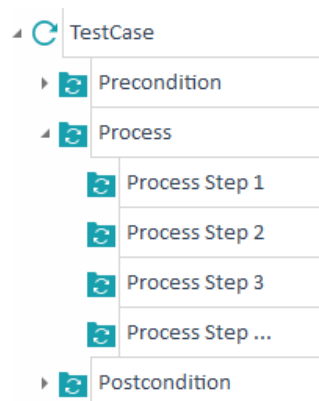
The next picture shows a recommended folder structure within a TestCase. Often not all folders are needed, especially not for simple TestCases. There is also an example for Recovery Scenarios on the Folder Level. These Recovery Scenarios will be used for all TestCases on the same level and beneath.



The content of these folders should be:

- Preconditions: Cleanup, Prepare data, Log in
- Process:
 - The process you want to test – should conform to only one topic
 - Verification of the expected result
- Postconditions: Release resources, Cleanup, Log out & close application

When automating End to End (E2E) TestCases, the structure within a test case needs to be adapted to fit a more complex workflow:



The content of these folders should in that case be:

- Preconditions: Cleanup, Prepare data, Log in
- Process Step 1-n:
 - Each process step may cover a different application
 - Should conform to only one topic each
 - If applicable: verification of the expected result
- Postconditions: Release resources, Cleanup

7.1.2.1 When to use Precondition & Postcondition

Use Precondition if you:

- Need to set buffer values
- Need to start the application
- Need to log in
- Need to navigate to a specific screen in the application where the process steps start
- Need to read data from Test Data Service (TDS) prior to test execution

Use Postcondition if you:

- Need to log out of an application after the test
- Need to close the browser or the application after the test
- Need to delete files that have been created during the test case
- Need to update an entry in Test Data Service (TDS) after the test process is completed

7.1.3 Naming convention of TestCases

Every Test case **must have** a unique name, e.g.:

TC_“Application Name”_BusinessObject_0001_“Purpose of Test case”

ApplicationName_BusinessObject_0001_PurposeOfTestCase

Within a Template, to create a unique test case name depending on certain values, the property InstanceName on the TestCase can be utilized. Please see an example below:

Information about “System” within TestCaseDesign:

Name	Debit Security...	Credit >1000	Credit Securit...
Transactions Security	Debit Security ...	Credit >1000	Credit Security...
Instances			
Administration			
Precondition			
System	SIT	UAT	UAT
Process			
Verification			

Property on the TestCase or TestCaseTemplateInstance Folder:

InstanceName {XL[Precondition.System]} | {XL[Instance.Name]}

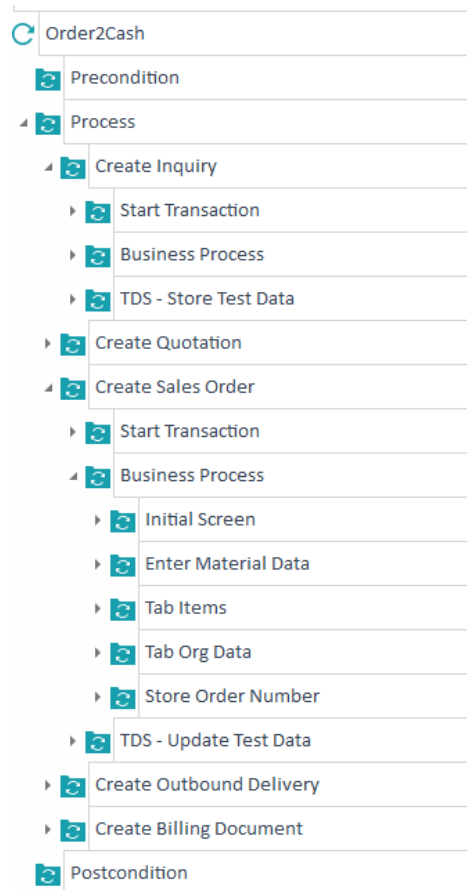
Final TestCase Name:

TemplateInstance of Transaction Security	
SIT Debit Security <=1000	
UAT Credit >1000	
UAT Credit Security <=1000	

7.1.4 Naming convention of TestStep Folders

TestStep Folders are a powerful tool to allow effective structuring of a test case to improve readability and process flow understanding.

A TestStep Folder should always be named according to the purpose of the containing steps.



Taking from the sample above:

- Create Inquiry: All the steps required to create an inquiry
- Start Transaction: All the steps required to enter a certain functionality
- Business Process: All the steps required to fulfill the purpose of this functionality
- Tab Org Data: All the steps that will be executed to handle the tab "Org Data"

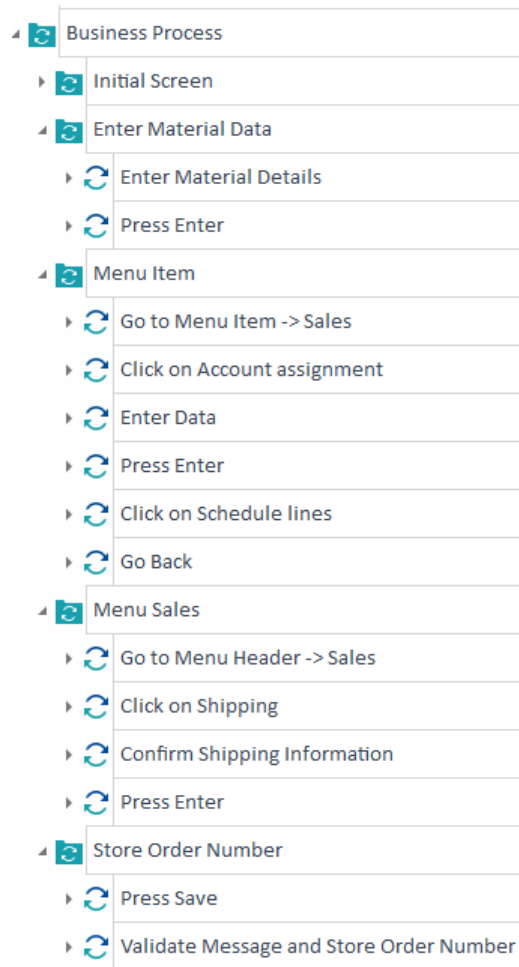
7.1.5 Naming convention of TestSteps

Every TestStep must have a descriptive name that allows for immediate understanding of its purpose. An easy way to achieve that is to apply a methodology like "Action -> Location". "What is this step doing" -> "on which object". Ideally, one TestStep fulfills exactly one functionality.

However, in some cases, it makes sense to have multiple functionalities in one step, e.g. on a PopUp.

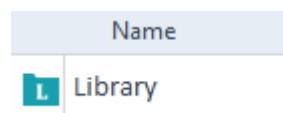
Examples:

- Press Enter
- Click on Schedule lines
- Press Save
- Enter Material
- Verify Message Type
- *On a popup:* Verify Success and Press Enter
- ...

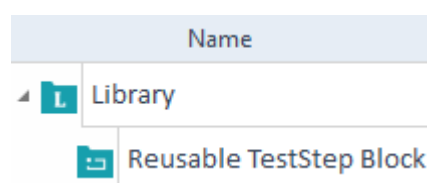


7.1.6 TestStepLibraries

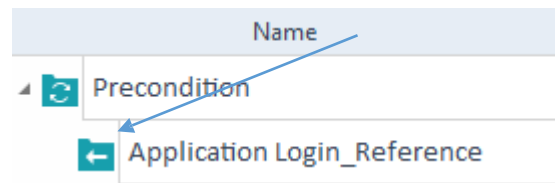
If a specific test sequence is used in many TestCases (e.g. Login process, etc.), we recommend using reusable test step blocks, which are stored in libraries. A typical example for a reusable test step block is a login procedure. The login will be used in many test cases and if something changes, it is less effort to integrate those changes at one centralized location. All usages are only linked to the library and can only be changed if the library is checked out.



A TestStepLibrary is the collection of ReusableTestStepBlocks.



These ReusableTestStepBlocks can simply be used with drag and drop on a TestCase. A reference to the ReusableTestStepBlock is created.



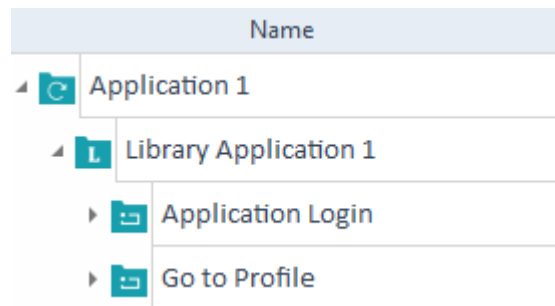
A change in the sequence of these steps can now easily be modified in the TestStepLibrary – and will immediately update all TestCases regarding this ReusableTestStepBlock.

7.1.6.1 How to use a TestStepLibrary?

In big projects it is recommendable to use more than one TestStepLibrary – as the Checkout of the whole Library is necessary for changing any of the included ReusableTestStepBlock.

As best practice, you can go by the predefined rule:

- Only one Library should exist for a specific Application and a specific application version.



7.1.6.2 When to create a ReusableTestStepBlock?

Optimal candidates for creating a ReusableTestStepBlock will stand out, e.g. the Startup of an application or the confirmation of Pop-Up-Windows. As you do not have to enter specific values, these steps will be the same.

In general, whenever you think about copying a part of an existing testcase, that part is a candidate for a ReusableTestStepBlock.

It is also recommended to create a ReusableTestStepBlock when you need a specific structure in the test case area, e.g.:

- Buffer used to identify a control in the Module
- Certain steps that need to work together for fulfilling a function
 - o Wait until page is loaded
 - o Refresh page until certain status is visible
- Custom built function that should not be visible to users to enable easier use of reusability

7.1.6.3 When not to use a ReusableTestStepBlock?

It might be ideal to use the existing TestStepBlock. Ask yourself: "Can I use it without making changes or implementing business parameters (see next section)?" If not: "Is the required change necessary for the sequence, e.g. because there is a new mandatory field added?" In this case, you

may have to change the ReusableTestStepBlock anyway and can then use it as reference in your TestCase. If not so – “Is there a profit in creating another ReusableTestStepBlock?” Proceed making sure you choose a unique name for both – as it seems they might be similar and to avoid confusion for all other TOSCA users.

7.1.6.4 Using Business Parameters

Business Parameters help you to parameterize some values in the ReusableTestStepBlock e.g. each usage of the login – ReusableTestStepBlock can use different login data.

	Name	Value	ActionMode	DataType
Library				
Web Shop Login				
Business Parameters				
Url			Input	
Email			Input	
Password			Input	
Open Web Shop				
Url		{PL[Url]}	Input	String
UseActiveTab		True	Input	String
OpenUrl				
Login				
Email		{PL[Email]}	Input	String
Password		{PL[Password]}	Input	String
Remember me?		False	Input	String
Login		X	Input	String

Tip:

If using Business Parameters for a username/password login use a buffer to store username and password. Set the data type of the password to “Password” to mask the password.

Further information on using Business Parameters can be found in the online manual.

[Business Parameters @ Support Portal](#)

CAUTION: If you use Business Parameters for within your Library, it will affect the reusability of the Library. You must provide values for those parameters all the time.

7.1.7 Using fixed values

When using fixed values in TestCases (without TestCaseDesign), please provide these values at the top of the TestCase by setting a Buffer of the used values that might be used multiple times within the TestCase.

Name	Value	ActionMode	DataType
Log in to your account			
Set Test Data			
Email	test@tester.com	Input	String
<Buffername>			String
Login			
Email	B Email	Input	String
Password			Password
Remember me?			String
Login			String

7.1.8 TestCase Templates

TestCases with a similar sequence can be generalized with a Template and be dynamically created using an external data source. This process is called dynamic TestCase generation. To convert a TestCase into a Template select the option "Convert to Template" from the context menu.

There are two ways to use test data with TestCase Templates, either from TestCaseDesign, which is the preferred and recommended solution, or if needed from an Excel spreadsheet with the data arranged in a way so that it can be imported into the Template.

Further information on dynamic TestCase creation with TestCase Templates can be found in the online manual:

- [TestCase Template @ Support Portal](#)

7.1.9 Recovery and CleanUp Strategy

Recovery and CleanUp Strategies can be used for:

- Recovery Scenarios can be used on different levels (TestCase, TestStep, TestStep Value) to define TestSteps to allow a Retry in case of a failure. Depending on the level of the recovery scenario, either the TestStepValue, the TestStep or the TestCase will be retried after the recovery TestSteps have been executed.
- CleanUp Scenarios are triggered if a TestStep failed, and all Recovery Scenarios failed as well. It can be used to clean up the test data and the test environment after a failed TestCase.

To learn how to use Recovery and CleanUp Scenerios, please refer to the manual:

- [Recovery Strategy @ Support Portal](#)

7.1.10 Advanced verification

Color

Identifying colors of an object can be done via the HTML properties of each object. If an object has a color property it can be identified as shown below:

	Name	ActionProperty	Value	ActionMode	DataType
↺ ↻	Verify the color displayed for Previo...				
?	Yesterday's net settlements (Header)				String
?	locationBased-Net Sales Week Over Week Comparison (Header)				String
?	NetSales Icon - previous week	attributes_stroke	attributes_stroke == #666666	Verify	String
?	NetSales Icon - previous week text				String
?	NetSales Icon - past week	attributes_fill	attributes_fill == ffcc00	Verify	String
?	NetSales Icon - past week text				String

Alignment

The alignment of objects can be verified like the color. If an HTML object has a property for the alignment, it can be verified. Therefore, see the sample screenshot below:

	Name	ActionProperty	Value	ActionMode
↺ ↻	Verify the display			
↺	Card Sales Info			Select
↺	Visa			Select
?	#2		Visa	Verify
?	#2	align	align == left	Verify
?	#3		*%	Verify
?	#3	align	align == right	Verify
?	#4			Verify
?	#4	align	align == center	Verify
?	#5		\$*	Verify
?	#5	align	align == left	Verify

If there are no properties, the best way would be to create partially automated test cases. Create a screenshot of the current UI and provide it for the manual tester in a specific folder. Therefore, the tester can do these verifications after the execution.

	Name	Value	ActionMode	DataType
↺ ↻	Verify the alignment			
Ⓢ	Environment	Desktop	Input	String
Ⓢ	Directory	C:\Tosca_Projects\Screenshots\ATM Details\General Info	Input	String
Ⓢ	Filename	TC02_ATMConfiguration_GeneralInfo_ DATETIME .jpg	Input	String

7.1.10.1 Table steering

TBox table steering

More than one steering action can be done in one TestStep. For selecting a specific row proceed as in the screenshot:

Confirm Order			
Products		Select	String
3rd Album		Select	String
Price	InnerText == 10*	Verify	String
<Cell>			String
50's Rockabilly Polka Dot Top JR*		Select	String
Qty.	InnerText == 1	Verify	String
<Cell>			String
<Row>			String
<Col>			String

A manual tester would select the cell e.g. Select Row "Name" and there the 1st column. Therefore, it will work even if the row number changes. In order, select a row or column with a specific value in one or any cell and you can use the ActionMode Constraint.

For more examples and possibilities within a table visit our online Manual:

- [Online Manual](#)

7.1.11 Cross Browser testing

General information on how to do cross browser testing can be found in the manual:

- [Cross Browser Testing](#)

7.1.11.1 Test Scope

Which test cases should be chosen for cross browser testing?

It does not make sense to test all TestCases in each browser as there is no functional difference between browsers. You may test a handful of test cases across browsers to ensure that the basic functions are being displayed and are working in different browser types.

7.1.12 Timeout and Wait

There are many locations where a timeout can be set, e.g. static wait or a wait on.

A static wait can be done with the Module TBox Wait, where the test execution is paused for the time specified.

For a dynamic wait use the ActionMode WaitOn. The ActionMode WaitOn pauses the execution of a TestCase until the respective control provides the value or property specified in the Value field. If the test object does not adopt the expected state in the Maximum Wait Duration a timeout occurs and the test execution resumes.

Wait on specific object/attribute



[WaitOn](#)

Static wait operation



[TBox Wait](#)

Wait command in the properties of a Module

Within the Properties of a Module, WaitBefore and WaitAfter Steering Parameters can be added to wait a specific amount of time after the control was steered.

Synchronization settings

If the automation is facing troubles with synchronization and timeout, these configuration settings can be changed:

[Settings TBox Synchronization](#)

7.1.12.1 Use “WaitOn” and loops

If there are some troubles with WaitOn-TestSteps that are already used, make sure that you wait for the objects that you want to steer afterwards. In one example, we tried to select a table cell:

Loop Example				
While - Wait for Order to be loaded				
Condition				
Verify Order not existing in table				
Order Table			Select	String
\$3			Select	String
\$1	Exists	Exists == False	Verify	String
Loop				
Refresh Table				
Refresh		CLICK	Input	String
Confirm Order				
Order Table			Select	String
\$3			Select	String
\$1		CLICK	Input	String

Looping as long as the cell does not exist

Refresh to load orders

Click on a order

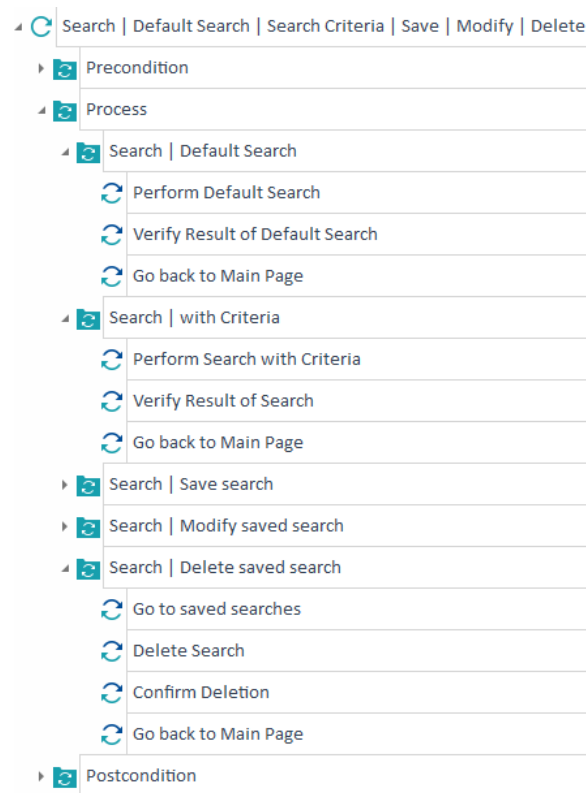
- [Conditional statements and loops](#)

7.1.13 When to split TestCases

It is not always clear at the beginning how to best split test cases. Help yourself by using the following guidelines to encircle the solution:

- ✓ Time-consuming processing (e.g. overnight or batch required)
- ✓ TestCase covers more than one topic (E2E Test vs Component Test)
- ✓ TestCase covers more than one requirement: can easily be linked to numerous requirements!
- ✓ Complex workflow for a business process including different user roles – think about splitting the TestCase and assemble the workflow in a Business TestCase
- ✓ 4-eye-check (logging in different users can easily be handled in one TestCase)
- ✓ Validation (check update of balance – may be required to hit the test target)

Example of a TestCase that should be split:



This TestCase covers more than one topic (Default Search, Criteria Search, Save, Modify and Delete Search). It should be split into smaller TestCases that cover one topic each. One way to split the TestCase would be:

- Search | Default search
- Search | Criteria search
- Search | Save search
- Search | Modify search (→ create and modify the search or depend on the Save search TestCase)
- Search Delete search (→ create and save the search to delete depend on the Save search TestCase)

Now each TestCase would cover only one topic.

7.1.14 Conditions

To make multiple instantiations of TestCases more flexible, the following objects can be linked with business-based conditions:

- TestCase Template
- Any XTestSteps within TestCase Templates
- TestStepFolders within TestCase Templates
- TemplateInstances
- XTestStepValues

During instantiation, these XTestSteps or Folders are only included in the TestCases if they fulfill the defined conditions.

With the conditional instantiation, the inclusion of dialog sequences in a TestCase depends on whether the data source contains the corresponding value for the specific business object or attribute in the Template.

This minimal difference in the dialog steering can be mapped in a Template as well.

Objects that are labeled as conditionally instanced change their symbol. TestCaseFolders are an example. Please see further details in the [online manual](#).

7.1.15 Error Handling done right

When Test Execution fails, Tosca supports you in several ways to find out whether there is a real error or if the failure appeared for technical reasons. The first indications are given already in the ExecutionList.

1. Search for the first TestStep in the execution that shows failure.
2. Expand failed TestStep and analyze Details / Loginfo (dependent on settings, make sure that you optimize this results)
 - ➔ Is it a validation error? Expand Details to find expected and actual result
 - ➔ Is it a timeout failure? Is it justified (expected time exceeded)? Check Settings
 - ➔ Application not steerable? Check manually - maybe a Pop-Up-window appeared, the reason is not always caused by technical changes
 - ➔ Follow the passed TestSteps manually and check, if the failure appears as well
 - ➔ Does the TestStep itself or the context clear up the reason for failure, e.g. when considering the action mode? Make sure that it is correct (input, insert, do-nothing, verify, constraint)

Consider the following benchmarks, which Tosca can easily inherit:

- Disable any controls that cannot currently be used. Once the control is released within the synchronization timeout, it is steered as intended.
- HTML applications: set the availability of the document accordingly. Tosca will then wait automatically prior to executing the next control.
- HTML applications: if the "PageSync" function is enabled, Tosca automatically waits for responses to the XmlHttpRequests (Ajax) sent in the background, until it proceeds with automation.

Fallback options:

- Steering not possible / changed technical ID – new Scan / ReScan
- CustomControl must be recompiled (e.g. upgrade TestSuite)
- Create manual TestStep until technical fix is available

Helpful Options & Settings:

Following Options and Settings can be enabled to ensure easier analysis of a problem:

Option "Create Logs for TestStepFolders"

Enabling this Option allows to view the complete test case structure within the Execution Log.

Options

General

Path Variables

Compact Workspace

Advanced

View

TestCase

ExecutionList

Hints

Optimize Perspectives

Old Workspace Schema

Compact Workspace

Workspace Size

Exploratory Testing

QTestIntegrationAddIn

RemoteControl

ReportingUI

RequirementsManagementUIAddIn

SAPSolutionManagerAddIn

TestCaseDesignUI

API_OSV Early Access

Tosca Data Integrity

OSV AddIn

Versioning Integration

Temporary Project Name	ToscaCommander
Common Repository access timeout (seconds)	5
Owned File availability timeout (seconds)	300
Search mode	Advanced
Initial restriction of search results	1000
Enable Business TestCases	<input checked="" type="checkbox"/>
LDAP authentication	<input checked="" type="checkbox"/>
Discard entries merged from TestMandates together with log	<input type="checkbox"/>
Enforce TBox Execution	<input type="checkbox"/>
Create Logs for TestStepFolders	<input checked="" type="checkbox"/>
Use alternative MSExcel interface	<input type="checkbox"/>
Return Licenses on Idle Timeout (Minutes)	-1

OK

Cancel

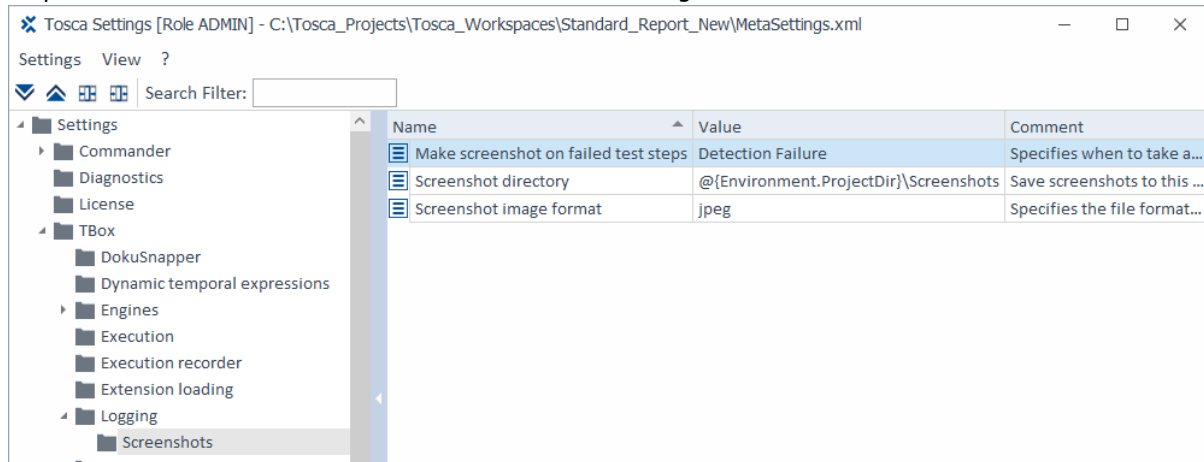
Apply

Details	Test configuration		
Name	Value	ActionMode	
▶ Create Sales Order			
ActualLog			
▶ Create Sales Order			
27.04.23 22:25:16			
SAP Logon			
▶ SAP Login			
User	test01	Input	
Password	*****	Input	
Language	EN	Input	
Enter	x	Input	
go to VA01			
Transaction code	/nVA01	Input	
Buttons	Enter	Input	
Fill Initial Screen			
▶ Press Enter			
Buttons	Enter	Input	
Enter Material Details			
Press Enter			
Go to Menu Item -> Sales			
Click on Account assignment			
Enter Data			
Press Enter			
Click on Schedule lines			
Go Back			
Go to Menu Header -> Sales			
Click on Shipping			
Confirm Shipping Information			
Press Enter			
Press Save			
Validate Message and Store Order Number			
▶ Close SAP			
Transaction code	/nEX	Input	
Buttons	Enter	Input	

Details	Test configuration		
Name	Value	ActionMode	
▶ Create Sales Order - Folder Structure			
ActualLog			
▶ Create Sales Order			
27.04.23 22:26:38			
Precondition			
Login to SAP			
SAP Logon			
▶ SAP Login			
User	test01	Input	
Password	*****	Input	
Language	EN	Input	
Enter	x	Input	
Process			
Create Sales Order			
Start Transaction			
go to VA01			
Transaction code	/nVA01	Input	
Buttons	Enter	Input	
Business Process			
Initial Screen			
Fill Initial Screen			
Press Enter			
Enter Material Data			
Menu Item			
Menu Sales			
Store Order Number			
Press Save			
Validate Message and Store Order Number			
Postcondition			
Close SAP			
Transaction code	/nEX	Input	
Buttons	Enter	Input	

Setting "Make screenshot on failed test steps"

When this setting is active, Tosca will automatically make a screenshot in case a step fails. This step will be published in the "Details" column within the Execution Log.



Name	Value	ActionMode	Loginfo	StartTime	Duration	Detail
Create Sales Order - Folder Structure						0:1:0:0:1
ActualLog			1	27.04.23 22:24:47	00:11.471	0:1:0:0:1
Create Sales Order						
27.04.23 22:44:47			Execution will be continued with the next TestCase, because the error situation could not be recovered.	27.04.23 22:44:47	00:11.471	
Precondition				27.04.23 22:44:47		
Process				27.04.23 22:44:47	00:11.399	
Create Sales Order				27.04.23 22:44:47	00:11.399	
Start Transaction				27.04.23 22:44:47	00:11.398	
go to VA01			Could not find Window 'go to VA01' with the following properties: - Caption: *	27.04.23 22:44:47	00:09.241	

Be aware, that the screenshot taken shows all screens connected!

7.1.16 Using Configuration Parameters

Configuration parameters are useful for defining global values that can be changed on each Tosca object.

Here are some sample parameters that might be useful:

- TDS connection
- DSN
- Environment
- Browser

7.1.17 Project Settings

Project settings define settings for all users that are logged in the same repository. Within the project settings, every setting of the Tosca Commander can be defined. The project settings apply only if the repository is in use. If you open another workspace with the same Tosca Commander, the general settings are used.

Name		Value
TCProjectSettings		
Commander		
General		
Advanced		
<input checked="" type="checkbox"/> UpdateUsersFromCommonBeforeLogin		True
<input checked="" type="checkbox"/> UpdateUsersFromCommonIfMissing		True
TBox		
Recovery		
<input checked="" type="checkbox"/> On dialog failure		Recover
<input checked="" type="checkbox"/> On exception failure		Execute next TestCase
<input checked="" type="checkbox"/> On verification failure		Continue
Tricentis Services		

For more information about all the Settings visit our online Manual. 

[Project Settings](#)

 [All Settings](#)

7.1.18 Using Tosca Recorder

More about the Tosca Recorder can be found [here](#).

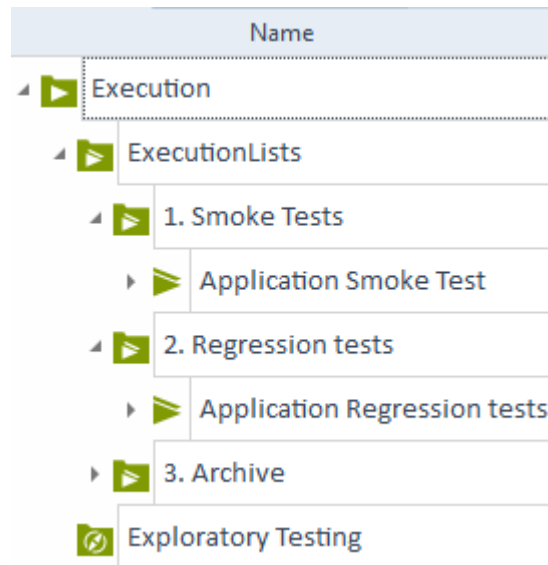
7.1.19 API Scan

More about the API Scan can be found [here](#).

8 Execution Section (Green)

The next chapters contain examples, lessons learned and tips from previous customer projects.

8.1 Folder structure

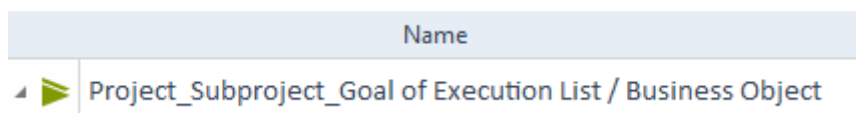


Once the TestCases are finished you can move on to the Execution section. Normally you will have two different ExecutionLists. One for Smoke test and the other on for Regression test. The Smoke test ExecutionList contains all high priority test cases all over your application. The Regression test ExecutionList on the other side will contain all TestCases that are supposed to be executed for Regression testing.

8.2 ExecutionList naming convention:

Every ExecutionList **must have** a unique name, e.g.

"Project"_"Subproject"_"Goal of Execution List / Business Object"



8.3 Folder structure in an ExecutionList

To make sure you can find your TestCases as easily as possible, you should keep the same folder structure you already have in the TestCase section. You can simply do that by copying and pasting from the TestCase section to your ExecutionList or by creating new execution entry folders.

 [Link to the online documentation](#)

8.4 Test Mandates

For an attended execution of TestCases, we suggest that you to break down the ExecutionList using Test Mandates. Especially in multi user repositories, Test Mandates are the easiest way to organize your test execution. However, the result of the execution in the Test Mandates will be automatically updated in the linked ExecutionList. All Execution Entries should be able to be executed without relating them to data preparation TestCases. If you have data preparation TestCases, please make sure that you execute them before executing Test Mandates.

 [Link to the online documentation](#)


8.5 Distributed Execution

To run many TestCases in parallel there are several options in Tosca:

- Jenkins to start test cases on certain machines.
- Distributed Tosca execution ([Manuals](#))
- qTest Integration

8.6 TC Shell

TC Shell is the easiest way to run scheduled TestCases automatically. TC Shell is a command line- based Tosca abstraction. Almost all features are available in TC Shell. To execute a scheduled TestCase automatically a batch file can start Tosca and the script with the TC Shell command.

Please have a look in the online manual regarding TC Shell: 

[Online Manual](#)

8.7 Exploratory testing

Within the Execution Section (green) in Tosca, Exploratory Testing Sessions can be managed. More about the Exploratory Testing in Tosca can be found [here](#).

8.8 Interactive testing

Within the Execution Section (green) in Tosca, Interactive Testing Sessions can be managed. More about the Interactive Testing in Tosca can be found [here](#).

9 Reports

9.1 Reporting

The easiest way to generate reports in Tosca is by pressing the printer button on the top right. You can print a report showing exactly what you see in your Tosca Commander section. This functionality can be used in all sections of Tosca, including Requirements and TestCases.

Details Test configuration						
Name	Value	ActionMode	LoginInfo	StartTime	Duration	
Create Sales Order - Folder Structure						
ActualLog			1	27.04.23 22:24:47	00:11.471	
Create Sales Order						
27.04.23 22:44:47			Execution will be continued with the next TestCase, because the error situation could not be recovered.	27.04.23 22:44:47	00:11.471	
Precondition				27.04.23 22:44:47		
Process				27.04.23 22:44:47	00:11.399	
Create Sales Order				27.04.23 22:44:47	00:11.399	
Start Transaction				27.04.23 22:44:47	00:11.398	
go to VA01			Could not find Window 'go to VA01' with the following properties:	27.04.23 22:44:47	00:09.241	
27.04.23 22:26:38			Testcase was executed on system:	27.04.23 22:26:38	00:13.609	

9.2 Requirement Set – Dashboard

The Requirements section in Tosca can also be used as a Dashboard. If you link TestCases with Requirements and ExecutionLists with the RequirementSet, you can easily track the current state of the System under Test.

Name	Weight	Contribution (%)	Coverage Specified (%)		Execution State (%)		
Insurance Application			80	20	66	14	20
Insurance Application	1						
Buy a Policy	1024	53,33	90	10	63	27	10
Life Ins.	256	42,66	100		67	33	
Vehicle Ins.	64	10,67	50	50	50		50
Handle Claim	256	13,33	25	75	25		75
Cancel a Policy	128	6,67	33	67	33		67
Provisioning	512	26,67	100		100		

9.3 Reporting Section

To generate more detailed reports, we recommend using the reporting section in Tosca. However, to use this section you must have a good understanding of the Tosca object module and how to use Tosca Query Language (TQL). Tricentis provides a couple of standard reports for our customers. Those reports are part of the "standard.tsu". If you want to learn more about the reporting section in Tosca, please read through the manual. ([Link to the online manual](#))

10 Best Practices



10.1 Do's and Don'ts


This chapter should provide some guidance on how to avoid common pitfalls. It contains several common scenarios with correct and incorrect examples.

10.1.1 TestCase Objects



These are general scenarios that apply to all types of objects in Tosca.


There are more than 15 elements on the same level. How should they be structured?

-  Put all elements on one level.
-  Create a new layer by creating a folder structure. Use meaningful names and group the elements accordingly.

 The number of 15 elements is not a “hard” limit – it is just a guideline. It might make perfect sense to create a new layer with fewer elements if the grouping is meaningful.



There are different types of objects on the same logical level. How should they be structured?


-  Put all objects in the same folder regardless of the object type.
-  Create a folder structure on that logical level and have only one type of object in each folder.

 Consider mentioning the object type in the folder name like “TestCase instances” and “TestCase Templates.”

10.1.2 Modules

The same screen is used in different places. How should the screen be scanned and organized into modules?

-  Create a separate Module every time to use that screen.
-  Create as few Modules for that screen as possible.

 There should be no Modules with identical structure in the repository, this means:

- same technical properties on the Module itself
- same ModuleAttributes

10.1.3 TestCaseDesign

When should you use the functionality “Generate Instances” or “Complete Instances”?

- ✗ Every time I am building a new TestSheet, to get all possible TestCases.
- ✓ As few as you can. You as a Test Case Design Specialist know the application better than anyone else. It is better to build all the Information you need for testing by yourself.

✦ At Tricentis we use this functionality in 5% of our projects. Most of the time we build it by ourselves.

When should you use the Relations in TestCaseDesign?

- ✗ Every time I am focusing a business rule in the application.
- ✓ If you use the automatic generation, you should treat it with caution. Otherwise, you never use it.

✦ Relations aren’t easy to maintain and have a huge impact when you use the automatic generation of instances. (See previous question)

10.1.4 TestCases

How should sequences of identical TestSteps that occur in different TestCases be treated?

- ✗ Repeat these sequences of TestSteps in each of these TestCases.
- ✓ Create Reusable TestStepBlocks for these sequences and use them in the TestCases.

✦ Reusable TestStepBlocks are centrally managed sequences of TestSteps that can be reused across different TestCases. A change to a ReusableTestStepBlock either in the Library or in one of the References will result in a change of all usages of that TestStepBlock.

Should TestCases be linked to Requirements?

- ✗ It is not necessary to link TestCases to Requirements.
- ✓ Yes, TestCases should be linked to Requirements.

✦ Only if linked to a Requirement, TestCases can be used to calculate the Risk Coverage. A TestCase can be linked to more than one Requirement.

***When using random test data, the state of the data might be different for each execution of a TestCase.
How can this be handled in a TestCase?***

- ✗ Use If-Then-Else during the process steps to decide which action to perform based on the state of the test data.
- ✓ Use If-Then-Else in preprocessing only to make sure the test data is in the correct state for the TestCase.
- ✓ Use Templates to provide test data.
- ✓ Use Test Data Management / Test Data Service (TDS) to create usable test data.

💡 The result of a TestCase shall be reproducible and it shall be possible to execute a TestCase again without manual test data preparation.

How should TestCases that consume data from other TestCases be grouped?

- ✗ They do not need to be grouped at all.
- ✓ They should be grouped in an ExecutionList.
- ✓ They should be grouped in a Business TestCase.

How do you show the readiness of a TestCase?

- ✗ Leave the TestCaseWorkstate value in its default state.
- ✓ On creation the TestCaseWorkstate defaults to PLANNED. As soon as you start work on a TestCase, set it to IN_WORK.

💡 TestCaseWorkstate is set to COMPLETED by QAS or TCD upon approval of the TestCase.

How do you properly synchronize your TestCase execution?

- ✗ Use static TC Wait module all the time.
- ✓ Use ActionMode WaitOn whenever possible on a particular test object/control.



💡 Static Waits should not exceed more than 5% of the total execution time of a TestCase.


10.1.5 ExecutionList

How should TestCases be grouped into ExecutionLists for Distributed Execution (DEX)?

- ✗ Use only one ExecutionList for all TestCases.
- ✓ Group the TestCases into a number of ExecutionLists according to the business work flow

How should TestCases be grouped into ExecutionLists on attended execution?

-  Use only 1 ExecutionList for all TestCases.
-  You have an ExecutionList named Smoke Test and Regression Test. For an attended execution you should group the TestCases into test packages by using of Test Mandates.

 Every Tester should get at least one Test Mandate for execution.

11 Resources

11.1 Web Links

 [Support Portal](#)

All Links mentioned in this document can be reached through our Support-Portal. The Support- Portal should be first point for every question you have.

 [Knowledge Base](#)

If you have a question regarding Tosca, please also check out our knowledge base. There you will find articles for other Tosca Users and our Support Team.

 [WebHelp Videos @ Tricentis Support-Portal](#) Interesting

videos on various topics can be found here