TRICENTIS

Test Design Specialist | Level Two

LESSON TRANSCRIPT

SURVEY

Tricentis Test Design Specialist | Level Two

Student Exercise Workbook

- Version 2019_04
- Designed to be used with Tricentis **Tosca version 12.x**

Lesson Transcript

This lesson Transcript provides the scripts used during the lesson videos for the Tricentis Test Design Specialist | Level 2 training.

Legal Notice

Tricentis GmbH
Leonard-Bernstein-Straße 10
1220 Vienna
Austria

Tel.: +43 (1) 263 24 09
Fax: +43 (1) 263 24 09-15
Email: academy@tricentis.com

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose without the express written permission of Tricentis GmbH.

© 2019 by Tricentis GmbH

CONTENTS

PREFACE

About this workbook

This transcript is specifically designed to supplement training of the Tricentis Test Design Specialist | Level 2

This transcript is divided into 1 Introduction and 1 Theory Lesson and 5 Lesson sections and supplies the script used for the lesson videos.

This transcript is not aiming to be a complete manual.

GETTING STARTED

GETTING STARTED

Introduction

Welcome to the Test Design Specialist Level Two course.

Tricentis Test Data Management helps you create, manage and provision the right data needed for testing. With on-demand access to stateful test data, testers can easily and quickly execute the most complex test scenarios.

Test Data Management is the umbrella term that we use for the holistic Tricentis solution. Its various components are grouped under Test Data Service, TDM Studio, qTest and Flood.

Test Data Service focuses on the test data life cycle and is the central test data hub. TDM Studio on the other hand, deals with the generation of data, masking production data or creating synthetic data. For more information on qTest and Flood, visit our website and register to the relevant trainings.

This course will focus exclusively on the Tricentis Tosca's Test Data Service and will equip you with the skills and tools you need to efficiently track, manage and monitor test data in a fast and organized way.
Nowadays, adopting a stateful approach is crucial for testing your system throughout the entire end-to-end chain. Tests often need to be executed in a sequence as the data has complex inter-dependencies. Without validating each business step, the test will produce inaccurate results as certain preconditions may have been missed. Testing the operation of a complex system therefore depends on how you manage the state of the data. This is particularly important when working in a distributed test environment.

In our Test Design Specialist 1 course, we focused on how to construct a data-driven test automation framework to include all the different specifications of the SUT. This enables us to create all the combinations of multiple TestCases that are required to ensure comprehensive test coverage.
What is the difference between TestCase Design and the Test Data Service?

Test Case Design can be thought of as a design mechanism, which structures TestCases strategically to include all the variants and pathways for the SUT. Some static data values can be added, but this data is non-consumable, that is it can be used multiple times and simultaneously.

Test Data Service, on the other hand, acts as a data repository, where data can be stored, retrieved and updated and its status easily viewed by multiple people. Furthermore, TDS stores consumable or dynamic data that regularly changes. This data can generally only be used once and by one machine at a time.

The question then arises, where do we acquire test data from? Before we dive deeper, lets first define test data.

Test data can be considered any kind of information fed into the application being tested. This could be a username, password, credit card numbers, customer details, reference data. The list goes on.

Nowadays, using test data for effective test automation poses many challenges for testers. The two main challenges being:

1. Acquiring specific test data
2. Ensuring that the state of the test data is efficiently tracked

Let’s take a closer look at the first point.

When working with a complex database structure, you will need to use specific test data. This external data might not available in your SUT or it may be difficult to access. Normally, an SQL query would be used to acquire the test data. However, it is often the case that SQL statements do not exist or are difficult and time-consuming to use.

Another scenario is if you already have test data available, such as production data, or data created by yourself from an earlier test or another system. However, retrieving this data is time-consuming.

In both cases, Tosca’s Test Data Service can be used to easily generate necessary test data, or, store and adjust available test data for your SUT with fast response and run times.

Now, going back to our second point on keeping track of the state of the data. In a typical testing environment, it is often the case that an earlier test has changed the data in some way. If one TestCase consumes or adjusts test data, it is important that inaccurate, old or consumed data is not reused in another TestCase. This particularly poses an issue if the end-to-end TestCases need to run across several days.

For instance, consider a simple scenario of logging into SAP with a specific user account to modify their personal details. Two TestCases cannot use the same user account at the same time as it would produce inconsistencies to the changes made in their personal details. Only once the first TestCase is executed and the data is consumed can the second TestCase run using those updated values.

Consider a more complex example of creating a customer account for a bank, creating an account number and processing a payment.

This three step process requires generating a specific customer ID, generating an account number and retrieving both of those dynamic data values to process the payment.

When running the TestCase on multiple machines, it would be vital to update each data entry progressively in the TDS repository before it is used again. For instance, it would not be possible to process a payment for a customer without having a dynamic customer ID number or account number available.

It is therefore imperative to keep the data stateful and have a clear overview of consumed data as it passes from one step to the next. Without this in place, wrong data will be retrieved and the test will produce inaccurate results, consuming time and money to analyze the point of failure.

Tosca’s TDS easily solves this issue. By adjusting the status, you ensure visibility in a multi-tester environment. You can also ensure that consumed data does not conflict with existing Testcases through lock and unlock features.

An additional benefit with TDS is that data can be created, retrieved or updated through API calls which has considerably increased the response time as well as the agility when managing test data.

Theory Lesson・Introduction to Test Data Service

In this lesson, you will gain a broader understanding of how **Tricentis Tosca's Test Data Service** works. First, we will cover some basic terminologies of TDS and its interface. Secondly, we will examine how TDS is used in conjunction with Tosca Commander to easily create, retrieve, update and delete data. All of this, in turn, enables efficient, organized and stateful Test Data Management.

This lesson will also go through the customer order process using the Demo Web Shop, which will set the basic scenario for the remainder of the course.

Please note, this lesson does not include any exercises.

Let's provide some basic definitions of TDS.

Test Data Service is a server-based application which uses a web-based portal to display the test data in table-like structure. This web page, otherwise called the **Test Data Management page** or the **Test Data Object Viewer** can be accessed anywhere by typing the appropriate endpoint request in the browser window.

The **TDS repository** is the name assigned to the database. When clicking on that repository, you will be navigated to the **TDS type**, which displays the set of test records categorized by type, for example CustomerDetails or SampleofOrders. **Test records** represent the individual rows or the data entries in the table. Importantly, the **Status column** specifies what state the data is in after a TestCase has been executed.

The question arises, how are all these data entries and columns generated in the TDS repository? To achieve this, a connection must be established between Tosca Commander and Test Data Service through installation and configuration, which we will cover in Lesson One.

Once communication has been setup, a series of TestCases are created in Tosca to specify what you want to do with the test data. The common functions are to create, find, update and delete test data. Importantly, there are **TDS Standard Modules** available that allow you to quickly and easily perform these functions.

Next, when the TestCase is run, Tosca Commander instructs the SUT what to do and also, interacts back and forth with the Test Data Service. Essentially, Tosca sends and receives a series of endpoint requests to and from TDS such as creating, finding or deleting the data. These changes and updates are reflected in the TDS repository accordingly.

For instance, when creating a new data entry, this information is displayed in the **table-like structure** of the TDS repository.

When a specific data record must be retrieved, Tosca searches for it in the TDS repository.

Let's take a concrete example to better visualize this process.

Consider the three step process of creating a user profile, creating a file and then uploading a file on a website.

In Tosca, we create three separate TestCases:

1. The first TestCase will create a new user account. Once the TestCase is executed, this new data entry will be displayed in the TDS repository.
2. In the second TestCase, we instruct Tosca to find and retrieve the new user account data from the TDS repository. Then, go to the website, login with the retrieved account data and create the file. Once the file is created, the data is updated to a new state in the table-like structure of the TDS.
3. In the third TestCase, we retrieve the newly created file data and use it to upload the file in the website. The data record is then further udpated, with those modifications clearly visible in the TDS repository.

You can think of Tosca working like the Manager in a project, sending requests on what needs to be done. TDS is considered like the Reporter who provides a clear overview of the status of the project as well as the role of the Document Controller, who stores all the relevant project documents under appropriate categories.

It is important to note that, when testing this whole scenario, each and every process is interlinked with one another and therefore it is not possible to test them independently. For instance, it would not be possible to upload a file without having first created a user profile, or the file itself.

This is crucial when working in a distributed environment. Luckily, with TDS, we can lock certain data that we use, such as a newly created file and use this test data exclusively in our TestCase, meaning that no one else will have access to that data. We will cover this in more detail in Lesson 2.
Now, let’s move onto our test scenario for this course.

We want to test the customer order process in the DemoWebShop.

This requires

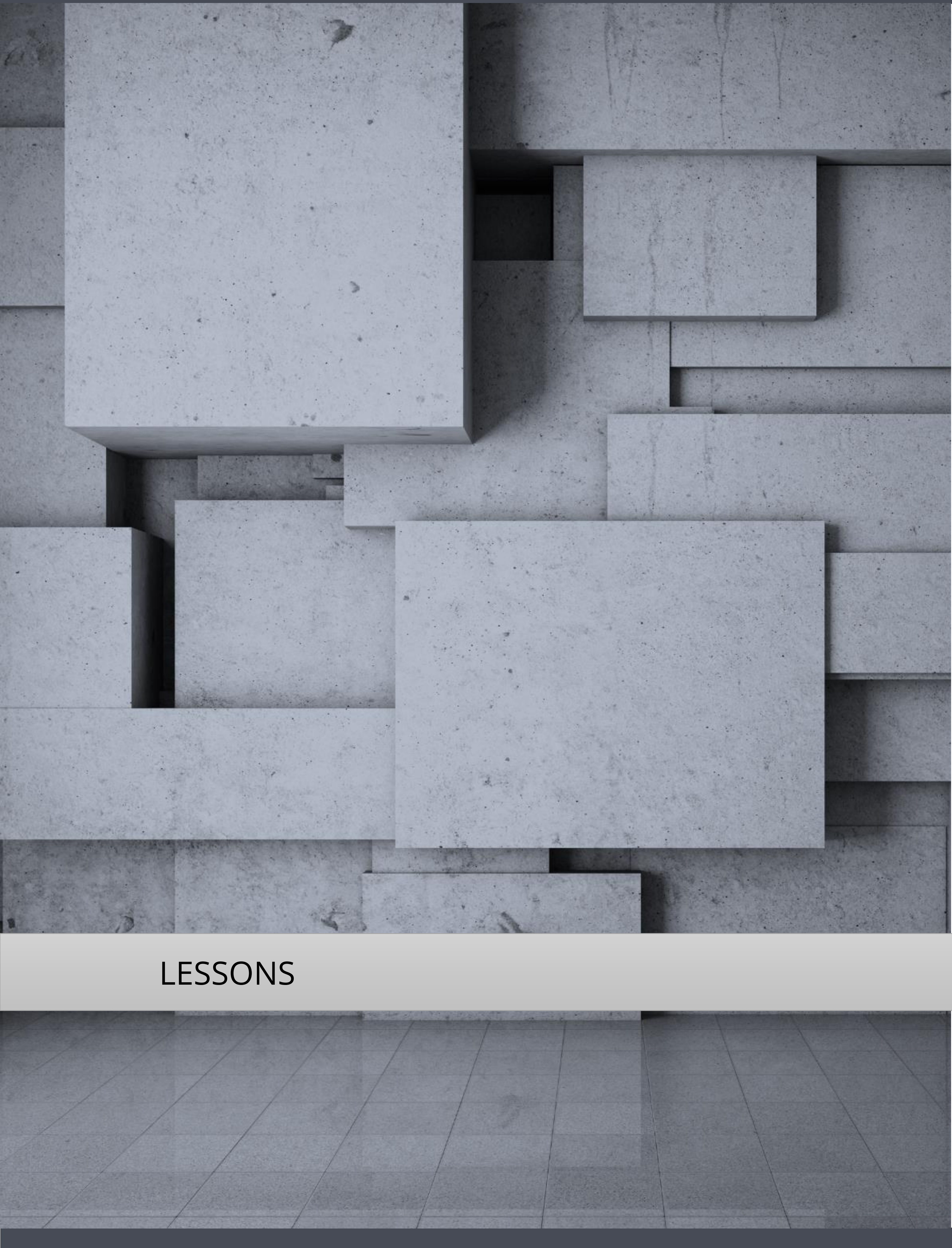
1. Registering a new customer
2. Adding an address
3. Adding items into the shopping cart
4. Purchasing these items
5. Processing the payment
6. Processing the purchase order

We will also test the process of cancelling an order and deleting the data record accordingly.

All of these steps require a series of updates and modifications made to the data. In Tosca, TDS Standard Modules are used to achieve this. After the TestCases are executed, the results of the data are reflected in the TDS repository.

For instance,

1. When a newly registered customer adds an address, the status of the customer is updated from New to Active in the TDS repository
2. Then, the customer will add items into a shopping cart, which will display a column called “items” with the number of items specified
3. Next, the customer will purchase items. This will create an OrderNumber column as well as an OrderStatus column set to Pending
4. Then, the customer’s payment will be processed through non-UI commands, which change the status from Pending to Processing
5. Next, in order to reflect the changes in the TDS repository, the OrderStatus record is updated from Pending to Processing
6. Finally, once the order is completed, the Test Records must be moved to a New TDS type called ‘Used’.



LESSONS

LESSONS

Lesson 1 · Preparing your Project for using TDS

In this lesson, you will learn how to setup Tosca Commander to connect to the **TestDataService (TDS)** repository hosted on the **Tosca Server**. In particular, you will learn how to:

- Install Tosca Server as well as the TestDataService (TDS) repository hosted on the Tosca Server
- Access the Test Data Management page which contains all the test data repositories and records
- Create a TDS repository
- Select and define a database type for the TDS repository
- Create and define TestDataService Configurations in Tosca in order to establish a connection between the Tosca Server and Tosca Commander

To start with TestDataService, there are certain **installation** and setup procedures that need to be followed.

For step by step instructions, please refer to the **Knowledge Base** article “Install Tosca Server” available in the Support Portal.

There is also a video guide available in the Resource Centre called “**Quick Set Up Guide**” which can be accessed through your Learning Management system.

First, you must download and install Tosca Server on the Support Portal which needs to be **version specific** to your Tosca Commander. For example, if you are using Tosca Commander version 12.0 then you need to download and install Tosca server version 12.0 respectively.

Then, once you launch the **Install Shield Wizard**, you will need to select the Test Data Service which comes as part of the standard installation process of the Tosca Server.

Make sure to also download the **TDS2 Base structure subse**t and save it in a folder from where you will import it later into your new project.

Once the set up and installation of the Tosca Server is completed, the webpage of the Tosca Server will automatically launch once the Install Shield Wizard successfully completes. This landing page contains all the features of the Test Data Service and the **Test Data Object Viewer icon** via one common web interface.

If you want to access this webpage, you must type the following endpoint request or syntax in the browser window: **http://<server address>:<port>**

- **<server address>** must be replaced with the name of the host or the IP address of the server. For example, localhost or <IP Address of Server machine>
- **<port>** needs to be replaced with the port number you specified during the installation. For example, 81.

Writing this port number into the browser window is only necessary if you changed the default port during the installation.

Finally, after following the steps outlined, you will be able to see the landing page, which will show an icon of the Test Data Object Viewer.

Click on this icon and you will be directed to the **Test Data Management page**. This page contains all the configured **test data repositories** and shows them as **tiles**.

Please note, when clicking on the Test Data Object Viewer icon, this creates an endpoint request in the web browser. The URL to the Test Data Management page will therefore be: **http://<SERVER IP or localhost>:<port-number>/TestDataObjectViewer**

To create a new repository on the Test Data Management page, click on the **Create Repository** button. In the subsequent dialog, you will need to define the following properties for creating a new repository:

- A ***Name*** Property – which provides a unique name of the repository you want to create.
- A ***Type*** Property – Tricentis' TestDataService supports the following types of Repositories:
    - ➢ Sqlite, which is a light weight Database that requires you to define the path of the location where the database .db file is saved. And
    - ➢ In Memory, where the data is stored temporarily in the computer's memory.
    - ➢ Here, you can select either the repository type from the dropdown menu. We will be using an SQLite database as the Repository Type for the purposes of this training.
    - ➢ <u>Note</u>: The TDS type name is not allowed to contain the following special characters: **/ ? # [ ] @ ! $ & ' ( ) * + , ; = \ :**
- A ***Location*** Property – where you have to define the path to the location where the database .db file is saved. The default location is %PROGRAMDATA%\TestDataService\<reponame>.db
- A ***Description*** Property – where you must enter a description of the new repository with a maximum of 200 characters

Finally, click on the **Save** button, which will automatically create a new database repository where all your test records will be stored.

The next step is to establish a connection between the Tosca Server and Tosca Commander. This requires defining the **TestDataService Configurations** in Tosca Commander.

To do this, first, you must **import** the TDS2 base structure.tsu subset to your new project workspace. Next, navigate to the **Configurations folder**.

Then, select the TestDataService configurations and define both of the following test configurations parameters:

- The **TestDataEndpoint** refers to the http location from where you want to retrieve your test records and
- The **TestDataRepository** refers the TDS repository from where test data will be queried.

Once the TestDataService Configurations are defined, you need to drag and drop them onto the relevant folders where those parameters will apply.

In our project, the TDS configurations will apply to all of our test cases and execution lists. Therefore, we drag and drop them onto our TestCase root folder and the required Execution List folder respectively.

For the sake of convenience, it is also possible to drag and drop the TDS configurations onto the Component Folder, only because those configurations apply to all of our test cases.

However, bear in mind, in real project scenarios, we may need to interact with different repositories for different TestCases. This would require different configuration parameters for individual TestCases. It is therefore best practice to drag and drop empty configuration parameters on the required TestCase folders and define their values separately for those TestCases. This would also be repeated for individual Execution list folders.

If you had a test scenario where the same TDS configuration parameters applied to all of your test cases and Execution Lists, on which objects in Tosca would you drag and drop those parameters?

The answer is:
Either on the Component Folder to apply those parameters on all objects, or on the Test Case root folder and relevant Execution List folders.

KeyPoints:

- Type the host or the IP address of the server into the browser window to access the Test Data Service repository
- Click on the Test Data Object Viewer icon to access to the Test Data Management page where you can create new repositories and view available records
- Click on the Create Repository button to create a new repository and define the set of properties in the dialog
- Drag and drop TDS configurations onto the relevant objects in Tosca

Lesson 2 · Create and Register records through TDS

In this lesson, you will learn how to use Tosca to **create** or **register new test records** into the TDS repository using **TDS Standard Modules**. You will also learn how to view these newly created test records in the Test Data Management page.

So far, in our Automation Specialist 1 and 2 courses, we have created TestCases to open the DemoWebShop, go to the registration page and register customer details. Now, we want to store this customer data into our TDS repository.

To do this, we create a TestCase and use TDS Standard Modules. These Modules are located under **Standard Modules -> Test Data Management -> Test Data Service**.

First, we need to create a TestStep and add the TD**S Standard Module called Test Data - Create & provide new record**. This is used to create and add the test records into the database of the TDS repository in a table-like structure.

Here you can see an example of a TestCase that creates a new test record through the TDS Standard Module.

When using the Standard Module, make sure to expand the **Data structure** Attribute and enter all the required Attributes and values which are later stored in your selected TDS type.

The TestStep Attribute **Existing or new TDS type*** is used for selecting an existing TDS type or creating a new type.

Alias name is a name assigned to specify a certain record in a dataset. This field is **optional**. If left empty however, the TDS type will automatically be used as an alias name. For example, the TDS type 'Customer Details' will be used to specify a test record.

It is important to know that the main purpose of using an alias name is to distinguish between record types. For example, a customer could be a sender or a recipient in a bank transaction. These two different types of users carry out different tasks in the SUT and therefore need to be tested separately. However, if no alias name were assigned to these different user types and they were all grouped under the same TDS type, the Test Case would fail as it would test the transaction process for a sender when a recipient would be required. Alias name is therefore relevant, especially when differences exist between the data records which perform different functions in the SUT.

In this course however, we will not be using Alias name for the purposes of this training.

To recap, have a look at the statements below. Which ones are true or false?

1. If the Alias name is left blank, it will take the name of the Existing Attribute
2. Alias name must be used when you have different types of records, which perform different functions in the SUT

The answer is:

1. If the Alias name is left blank, it will take the name of the Existing Attribute (**False**, it takes on the name as the TDS type)
2. Alias name must be used when you have different types of records, which perform different functions in the SUT (**True**)

Once the attributes and values have been entered, the TestCase is run on Scratchbook which registers all the data in your test data repositories.

To view all these test records, you must access the Test Data Management webpage.

This landing page shows all the configured TDS repositories as tiles.

After navigating to your respective repository, you will see a Types column on the left side. The names in blue display all the different TDS types that you created by the TDS Standard Modules in your TestCase.

The number of records created in that TDS type is shown in a small green box.
The Test Data Management page displays all the records of the selected type on the right side of the page, in a table-like structure.

After you have found and retrieved a record, it becomes automatically locked. This means, no one else in a distributed working environment can use that test record in a TestStep. It is only after the test has been executed that the record will become automatically unlocked. This ensures that there are no discrepancies created in the data when working in a distributed environment.

Now that we know how to register test records into the TDS repository, the next step is see what values to input in a TestCase in order to register specific test records into the TDS repository. Here we will use an example from the demo web shop.

Once again, the TDS Standard Modules: **Test Data - Create & provide new record** is used for registering the test records in the TDS repository.

Some data from the demo web shop, such as first name, last name and email, have been saved as Buffers and these buffered values will be used as inputs in the standard module.

The **Status** with the value **New** indicates newly registered customers. This Value will be used later to retrieve those customers in that state.

The values of the Attributes OrderNumber and OrderStatus are also required for later use when retrieving customer data. The values have been left blank as they will be progressively updated as the data is retrieved. For example, once an order number is generated, this data point will be updated and then moved.

So far, we have been running our tests in the Scratchbook for registering the data in our TDS repository.

Please note however, that in a real project scenario, we would have to create an ExecutionList and execute our tests there. This would ensure we have clear reports and overview of the test records that we generate and update, creating visibility in a distributed work environment.

As we would like to **generate multiple test records** in the TDS repository, it is necessary to specify a **Repetitions** value on the relevant **Execution Entry**. To do this, you must navigate to the relevant Execution Entry, and from the Context Menu in the Ribbon pane, select Repetition, specify the value in the Replace with field and enter Modify All. For example, if you enter a value of 5, this will generate 5 test records in your repository.

Alternatively, the Repetition value can be altered in the **Properties** pane when selecting the relevant Execution Entry.

Key Points:

- Create test records using TDS Standard Modules
- View test records in the Test Data Management page

Lesson 3 • Retrieving and Updating Test records through TDS

In this lesson, you will learn how to perform three main functions in the TDS repository: First, you will learn how to **retrieve** existing test records from the TDS database, second, **update** these test records, and finally, **move** these test records from one TDS type to another. In addition, you will also learn how to use the TD expression whose function is to enter retrieved test records into the DemoWebShop.

In our scenario, we will test the following customer process in the DemoWebShop:

1. First, A customer registers their email in the DemoWebShop
2. then, the newly registered customer adds an address
3. Then, the customer adds items into a shopping cart
4. Next, the customer purchases items
5. The customer payments are processed
6. Next, the purchased order is processed
7. And Finally, the order is completed

It is important to note, that when testing this customer process and structuring your TestCases in Tosca accordingly, the series of changes made to the test records will be reflected in the TDS repository.

For example, when a newly registered customer adds an address, their status changes from new to active in the table-like structure of the TDS repository.

Now, lets look more closely at how to find, update and move test records.

<u>Retrieving test records:</u>

The first step is to find or retrieve the correct test record that you created earlier in the TDS repository.

To retrieve a test record, you will carry out a search query using the TDS Standard Module "**TestData - Find & provide record**".

When performing the search, we must first specify the **Existing TDS type*.** This defines the type of dataset where the data record is located, for example, customer details.

In order to restrict the search to a specific set of records within the TDS database, the **ActionMode Constraint** is used. In this example, the search is restricted to the TDS attribute Status with New as its value. This means searching for any newly registered customers in the demo web shop.

You'll remember from Lesson 2 that we did not use an Alias name to specify a record in a dataset. This is because we are testing one type of user, and that is, a customer who registers in the demo web shop and performs the order process.

As no Alias name was provided, Tosca will automatically use the value provided in the **Existing TDS type*** to search for that record. In this case, Tosca searches for any record within the Customer Details dataset type with the Status 'New'.

<u>Entering the retrieved test records:</u>

After the correct record has been found, we now need to use that retrieved data and input it into the demo web shop. This means, inputting any newly registered customer's Email to login to the Web Shop.

To do this, we use the **{TD[...]} expression**, which should be familiar to you. It has the same construction as other expressions you will have seen when completing the Automation Specialist courses.

The {TD[...]} expression is entered into the Value field for a **TestStepValue**.

The first part of the 'TD' expression, directly after the curly bracket, is a command which asks the system to retrieve test records from a specific TDS repository.

The second part of the syntax, in the square brackets, defines exactly which record you wish to retrieve and use. This is done by specifying the TDS type first followed by the TDS Attribute, separated by a full stop:

{TD[ObjectName.AttributeName]}

For example, in the context of this training, if you would like to use a newly registered customer's email address and input it into the demo web shop, what would be the correct TD syntax to enter into the Value field?

The answer is:

{TD[CustomerDetails.Email]}

This essentially tells the system to retrieve a record using the TDS type "CustomerDetails" and use the Email address provided in the TDS Attribute called "Email", a command which will enter this data in the demo web shop.

<u>Updating a test record:</u>

After a newly registered customer's address is added into the demo web shop, we now need to change the Status of that customer from New to Active.

In order to do this in Tosca, that is, update the test records in the TDS repository,

we use the TDS Standard Module "**TestData – Update record**".

Remember, to update the test record successfully, we first have to find the correct test record from the TDS database and then update it. Essentially, updating test records in TDS database always works in conjunction with a search query.

When updating the test records in Tosca using the TDS Standard Module, the value for the TestStepValue "**Existing alias name (record)***" needs to be the same as the value provided when searching for the record in that TDS dataset. For this example, the value for the "**Existing alias name (record)***" will be CustomerDetails.

To now update and change that record, the search criteria will use the TDS attribute Status, and update the value from New to Active.

After the TestCase is run, the changes, that is, the Status update, are reflected in the TDS repository.

<u>Moving Test Records to another TDS type:</u>

As we stated earlier, we are testing the order process in the demo web shop application using test records from the TDS repository.

After each step of the customer order process is completed, this creates a series of updates in the TDS repository. For example, after a customer adds items in the shopping cart, the number of items will be displayed in the TDS repository after the TestCase is run.

Similarly, after the items are purchased, the order number will be created and the order status will be set to Pending

Then, after the purchase order is processed, the OrderStatus will be changed to Processing in the TDS repository.

It is important to highlight that processing the customer’s payment will require a two Step process. First, we will use non-UI standard Modules to process the payment. This will involve searching for a record with a Pending OrderStatus in the TDS repository and using non-UI requests to enter this OrderID number into the backend of the system, along with the customer’s email and password. Tosca then verifies that a **200 OK response** is received, indicating that the customer’s payment has been successfully processed. This changes the status to Processing in the backend.

Secondly, in order to reflect these changes in the TDS repository, Tosca searches for the same record with the Pending OrderStatus and verifies that it has been set to Processing in the DemoWebShop. It then updates that same record to Processing in the TDS repository.

Once the customer has completed the full order process in the demo web shop and the payment has been processed, the data is considered consumed.

This therefore requires moving the used customer data into another TDS type. This ensures that the data is kept stateful, that is, the end-to-end testing process of the SUT is efficiently tracked. This also makes the process of managing the test records more efficient when working in a distributed team environment.

In order to move data from one TDS type to another, you will use the TDS standard Module “**TestData - Move record to TDS type**” in Tosca. This Module first looks for the test records, which in our scenario are located under the TDS type ‘CustomerDetails”. Then, those records are moved to a new repository specified under a new name, in our case ‘CustomerDetails_Used’.

In this example, the TestStepValue **Existing alias name (record)*** will take the value of the existing TDS type. For instance, ‘CustomerDetails’. However, the TestStepValue **Existing or new TDS type*** will take the value of the TDS type where you want to move the used customer records to. This could be ‘CustomerDetails_Used’.

Key Points:

- How to Retrieve and Update repository records using TDS standard modules
- How to Enter data from TDS repository into the Demo Web Shop application
- How to move used test records from current TDS type to a different TDS type

Lesson 4 • Introduction to the Standard TestData– Expert Module

In this lesson, you will learn how to use the **TestData - Expert Module**. This Module performs all the functionalities of all the other TDS Standard Modules that we have covered so far, such as **creating, finding and updating test records.**

The Expert Module also provides you with additional functionalities, such as the **ReadOnly** feature, deleting records and deleting types, which we will go into later.

So far, you have learned how to create, find and update test records in the TDS repository using stand-alone **TDS Standard Modules**. Rather than structuring a TestCase by using all these Modules separately, it is possible to use the Standard TestData Expert Module which combines all these different functionalities in one single module.

This Module is included in the **subset** Standard.tsu and located under **Standard modules->Test Data Management->Test Data Service**.

For example, we would like to test the process of ordering multiple Tricentis Hoodies and then cancel the order. In Tosca, the TestStep flow will appear as follows:

1. Create Multiple Orders
2. Read the Created Order
3. Find and Update the First Order to Cancelled State
4. Find and Update the Subsequent Orders to Cancelled State
5. Find and Delete the Cancelled Order
6. Delete the Entire TDS type

To structure the TestCase, first we add the Standard TestData Expert Module.

In this Module, you will see the TestStepValue **Test data task***. This contains a drop-down list of all the different functions that are performed on the test data, such as 'Create' 'ReadOnly' 'Find', 'Update', 'DeleteType' 'DeleteRecord' 'DeleteAll'. These properties essentially instruct Tosca on what you want to do with the data in the TDS repository.

Let's look at an overview:

- The **Create** property creates a new TDS type or adds values to an existing type
- **ReadOnly** will read the record and unlock it
- **Find** will search for the FIRST existing record based on the criteria
- **Update** will edit a single record's values
- **DeleteRecord** will delete a single record from the TDS type
- **DeleteType** deletes the entire TDS type from the repository
- **DeleteAll** will delete all contents of the repository

Let's take a look at each one of these properties more closely.

So far, we have not implemented the 'Read Only' function in our TestCases. The ReadOnly feature allows reading test data information without performing any modification to it. One important distinction to make is that it does not lock the data record. All the other functions lock the test data by default.

You might be asking under what circumstances would it be useful to use the ReadOnly function, that is, keep the data unlocked when working in a distributed test environment?

Some use cases of ReadOnly is if you would like to use data that can be re-used, such as:

- contract data
- user ID numbers
- reference data or
- credit card numbers when testing an application.

For example, when testing the bank transaction process, you may require any visa card number and it does not have to be unique. In this context, it is important that multiple testers can borrow this data so that several tests can run at the same time on different workspaces and work stations in a distributed environment.

However, if this data was locked by default, the TestCases could fail as the same test record could be retrieved in different TestCases when executing a test. The ReadOnly property and its function of unlocking data is useful in these scenarios.

Going back to our DemoWebShop example, we will now explain how to structure the TestCase using the Expert Module to complete the flow of the order process outlined earlier. That is, create multiple orders for Tricentis Hoodies and cancel the order.

To create the order, first we will add the Expert Module and select ‘**Create’** from the drop-down list for the TestStepValue **Test data task*.** Then, we will assign SampleOrders as the name for the TDS type. We will also assign the name ‘order’ for the Alias name (record).

Once the data structure TDS attributes have been entered, such as the OrderID, name of the Product, quantity, Price, Order Status and Year of Manufacture, these properties will be displayed in the TDS repository as separate columns.

The second step is to read the order we just created. After adding the Expert Module, we will select ‘Read Only’ from the drop-down list for the TestStepValue **Test data task*.** As mentioned, this will automatically unlock that test record, meaning that anyone could use the data record in their respective TestCases.

For the purposes of this training, we simply demonstrate the usage of the Read Only feature. However, in a real project scenario, the relevance would be to keep this record unlocked so that any order number.

Additional Lesson • Managing TDS repositories using Tosca API

In this video we are going to demonstrate how to use Tosca API for creating, retrieving, updating and deleting test records in the TDS repository.

So far, we have been managing test records using TDS Standard Modules in Tosca to update the Test Data Management page. Now, we will see how this is possible using the Tosca API scan.

We will add new repositories with the help of RESTAPI. To do this, we need to call an endpoint to add new repositories.

To do this, navigate to the API Testing tab and click on API Scan. Here you will see a folder. Rename it to "**Manage Repositories**". In the folder you will see a default message called "**Create Message**". rename it to "**Create Repository**". Now, we will select the Method POST, and in the Endpoint, we will insert the repository path. The Message format is the standard format which we must use to create a repository and enter the relevant values. First, copy and paste the below **Message format** onto the **Payload tab**.

Message Format:

```
{
"description": "First_Repository",
"location": "%PROGRAMDATA%\\Tricentis\\TestDataService\\test.db",
"name": "abc",
"type": "Sqlite"
}
```

On the right side of the screen, under **Headers**, it is necessary to enter the value for the property. For "**Name**" input the value "**Content-Type**" and for the property Value input the value "**application/json**".

Run the message and navigate to the Test Data Management page of the Tosca Server. After refreshing the Landing Page you will see the new repository we just created.

Now as we have created the repository, we will have to retrieve the repository details. To do this, right click on the "Manage Repositories" Folder and select "Create Message". Rename it to "Retrieve Repository" and then, select the method GET. Insert the repository path into the Endpoint. Please note, this will be the same repository path that we inserted while creating the repository). This will endpoint request will retrieve all the repository details. Now, run the Message. In the Payload tab you will see the repository details we just retrieved.

Now we will demonstrate how to update the repository. To do this, first right click on the "Manage Repositories" folder and select "Create Message". Rename it to "Update Repository" Then, select the Method POST and insert the repository path into the Endpoint. Copy the Endpoint from the above message and paste it into this Endpoint, also enter the repository name you wish to update. We follow the same standard format for adding values to update a repository as outlined previously. First, copy and paste the above Message format on to the Payload tab. Here you can make changes to the existing repository details. Once you execute the message, the changes will be reflected in the Test Data Management page of TDS. Ono the right side of the screen under the Headers, it is necessary to pass the value for the property. For Name input the value "Content-Type" and for property Value input the value "application/json".

Run the message and navigate to the Tosca Server. Refresh the page and on the landing page you will see the changes made to the specific repository.

Now as we updated the repository earlier, here we will showcase how to delete the existing repository. To do this, right click on the “Manage Repositories” folder and select “Create Message”. Rename it to “Delete Repository” and then select the method DELETE. Insert the repository path into the Endpoint which is the same repository path that we used when creating the repository followed by the repository name which you want to delete. This will delete that specific repository.

Run the message and navigate to the Test Data Management page of TDS. After refreshing it, you will see that the previous repository with the name “DemoWebShop” will have been deleted.

We can also save these artifacts. To do that, select “Save As” option from the menu bar. This will be saved in the **.tsu** format, which can be accessed anytime in the future by importing it as a subset in the Tosca workspace.