TRICENTIS

Test Data Management
The Underestimated Pain

Wolfgang Platz, Founder and CPO

According to our customer surveys and confirmed by industry statistics, manual testers spend 50 - 75% of their effort on finding and preparing appropriate test data. Considering the fact that manual testing still accounts for 80+% of test operation efforts, up to **half (!) of the overall testing effort** goes into dealing with test data.

In addition, **test automation** requires having test data under full control: only with test data perfectly fitting the test description can automated test cases obtain stability. This is less an issue for simple test scenarios that create business objects (such as *business partners*, *accounts*). But it becomes a huge challenge as soon as objects need to have a certain state or history in order to be an appropriate testing basis (e.g.: *accounts* with a certain balance which require a series of previous transactions).

Despite the critical role of test data management, organizations usually do not pay an appropriate amount of attention to it from the beginning. It's an underestimated pain.

50–75%

of manual testing effort is used to find and prepare appropriate test data

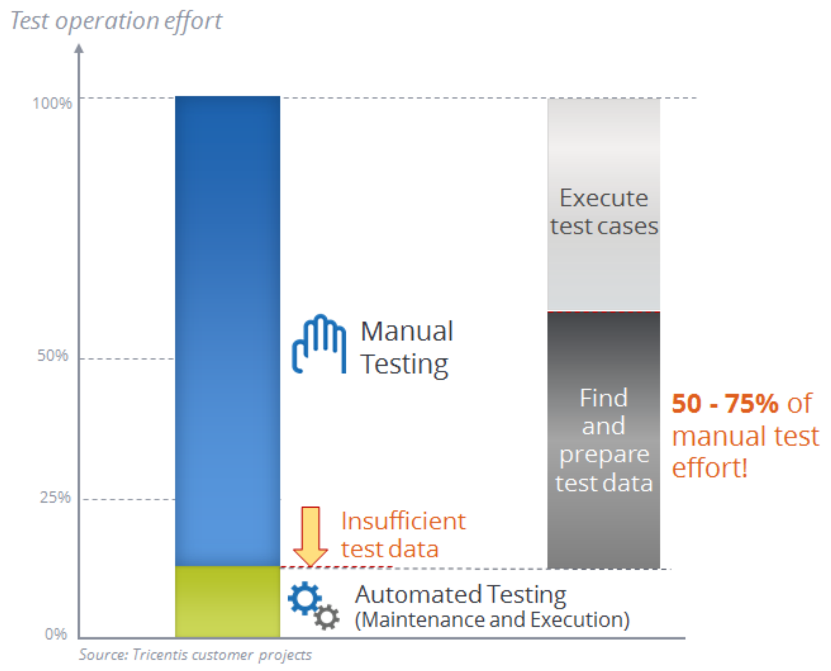
Why people love production data

Using production data (live data) for test purposes seems to be a simple solution and is the natural approach to fulfill the needs of testing. Customers expect from production data:

- Production data contains all test-relevant combinations for testing.
- Production data is already there!

That's a huge advantage! No need to create test data – just use them!

However, using production data for testing has some severe down-sides that massively constrain its usability, cause risk, and create massive efforts.



Graphic: Finding and preparing test data generates huge effort for manual testing + constrains test automation rates.

Why production data is insufficient, expensive and even dangerous

Let's have a closer look at production data and critique the previous statements. We will, however, find out that they are only partially true:

- Production data contains all relevant combinations for **regression testing**.

  While production data contains a great variety of data for regression testing, they are often not sufficient for new functionality. Since this functionality has not been productive as of yet, it is likely to miss test data for its test.

- Production data is already there! – but is **grows old**.

  Production data is usually provided as a snap shot. The problem with this: the data grows old! A test-person of age 17 may have turned 18 – and not be suitable for your particular tests any more. You would need to find a new person – or create one.

Operating test environments with **production data is expensive**: since the data is of considerable size, operating them over time consumes CPU which may cause substantial cost, especially for HOST systems (MIPS consumption).

Even more, the use of production data for test is becoming a no-go in enterprise companies, since ...

- **Using production data is a security issue.**

  Test data may be exposed to such unauthorized sources like in-house testing staff, consultants, partners, and even offshore personnel. Compliance regulatory has become very strict about the use of live data for test purposes, requiring anonymization (masking).

  For complex system landscapes, masking is a huge challenge that can hardly be accomplished or not solved at all.

69%

of IT professionals use production data for testing their apps[1]

*"A huge benefit of our **switch to synthetic test data** was the reduction of data volumes. All of a sudden our **operating cost of the test environment** – which was primarily driven by MIPS consumption **– went down by 95%**."*

QA director, major retail bank

$3.5 m

average cost of a data breach to a company[2].

[1]... The Insecurity of Test Data: The Unseen Crisis, Ponemon Institute

[2]... 2014 Cost of Data Breach Study: Global Analysis, sponsored by IBM

Synthetic data can be used to test both existing and new functionality – and it does not face any privacy issues, making it an attractive alternative.

Synthetic test data also contains thousands of records instead of millions, thus reducing your operating costs significantly.

Using synthetic test data is a challenge too.

On the other hand, the provisioning of synthetic test data is an effort that **requires full automation**. And synthetic test data needs to cope with the **volatility of date/time.** It requires a **detailed provisioning plan** to obtain the desired test objects and states.

How to set-up a sufficient test data management

Tricentis has come up with a **list of 5 steps** to shift test data management to the level you require:

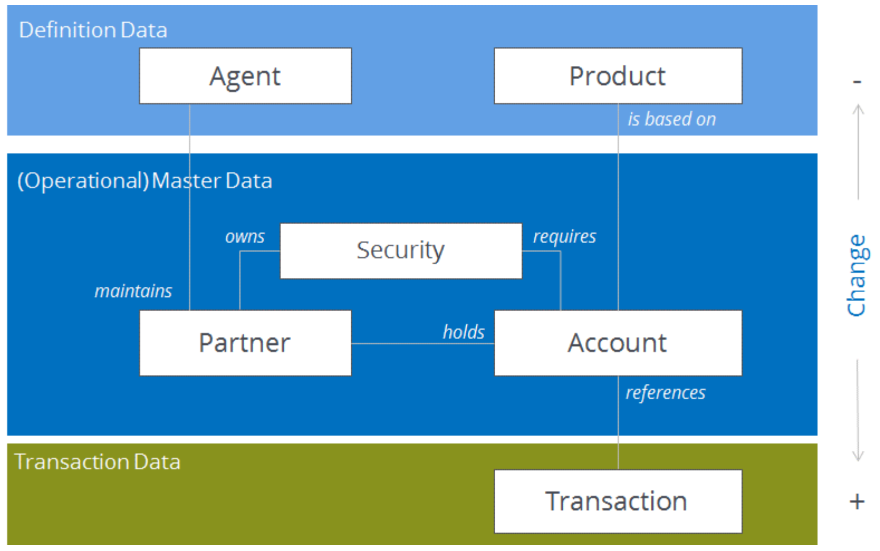
1. Scope out the area of active test data management
2. Automate provisioning of synthetic test data
3. Connect multi-step scenarios with a logical time axis
4. Complement remaining leaks through extracts of production data
5. Obtain "stateful" test data management

1

Scope out the area of active test data management

The first step is to find out which data objects we need to be able to create automatically whenever needed – those objects will to be under active control of test data management:

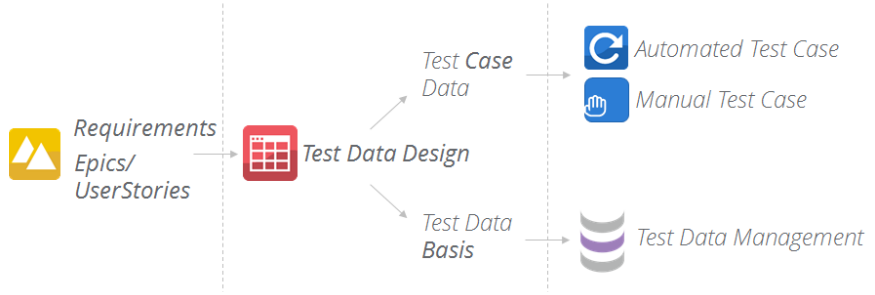
- **Definition data**: This data (e.g.: product definitions) does not change substantially from release to release, nor is it consumed by the test per se (a product does not vanish when it is used as the definition of an account). The number of records is limited. We do not need to actively manage them.
- **(Operational) Master Data:** This data (partners, contracts, accounts) is subject to ongoing change. Comprehensive tests will require thousands of records that we need to automatically generate and actively manage.
- **Transaction Data:** The most volatile data comes in high volumes and will affect operational master data. We need to automatically create transactions and track the changes they cause.



Graphic: Out of this simplified bank example, all operational master data and transaction data will be subject to an active test data management.

With Tricentis **Tosca Testsuite™**, customers design test case data simultaneously with the required test data basis.

E.g.: To create a certain *loan account*, the app must already have registered a *business partner* with a good rating (the future *account holder*). When testers design the test data for the *loan account*, they will also design the specific *business partner* that must be provided prior to the test.



Graphic: Test Data Design and Test Data Management in Tosca Testsuite™, process overview

2

Automate provisioning of synthetic test data

Data objects under active management require automated provisioning – otherwise you will end up with repeated pizza-party-style manual data generation events (your SMEs will refuse to do that) and never achieve sophisticated data object states (since those will require a series of steps to be performed).

In principle, all possible interfaces to the app's data layer are a feasible option for the automated generation, however, we recommend to use the app's APIs for this purpose:

- **UI based** test data generation is time-consuming (slow) and lacks stability.
- **Direct SQL insertion** into the app's persistency layer is certainly fast, but will have to rebuild parts of the app's business logic in order to achieve consistent data – leading to an SQL based development project.
- **Using an API interface** is the best option, since it is both fast but already benefits from the app's logic to interact with the persistency layer.

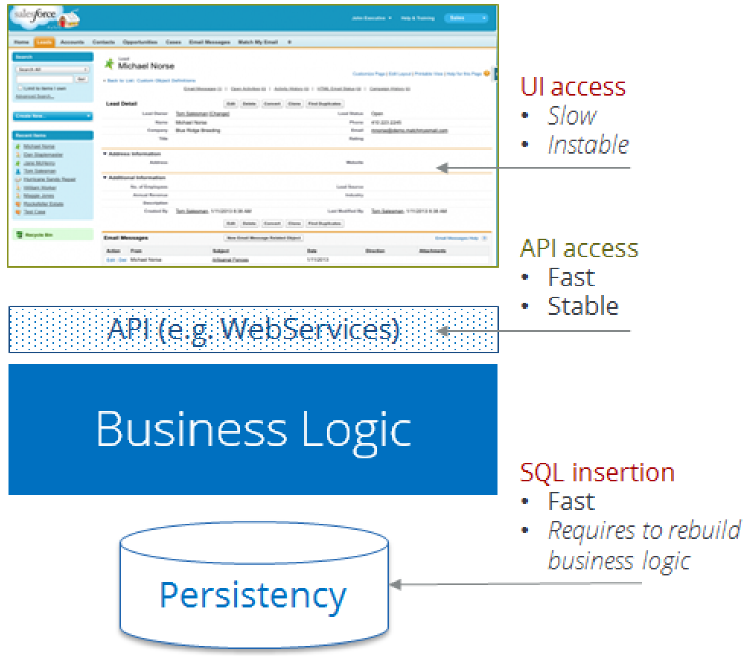
Bottom line: whenever the required data *can* be created/administered through an API, use it. **Tosca Testsuite™** is an ideal solution, since its **Model Based Test Automation** supports all three options.

Tip

SMEs of your business units might want to assign "their" names to *business partners*. It will be an easy thing to accomplish this wish and greatly enhance the acceptance of synthetic data.

"We switched to synthetic test data because of data privacy issues. Before Tosca, creating the required data was a huge manual effort that we had to go through repeatedly. Now it's all automated."

QA director, major Swiss bank



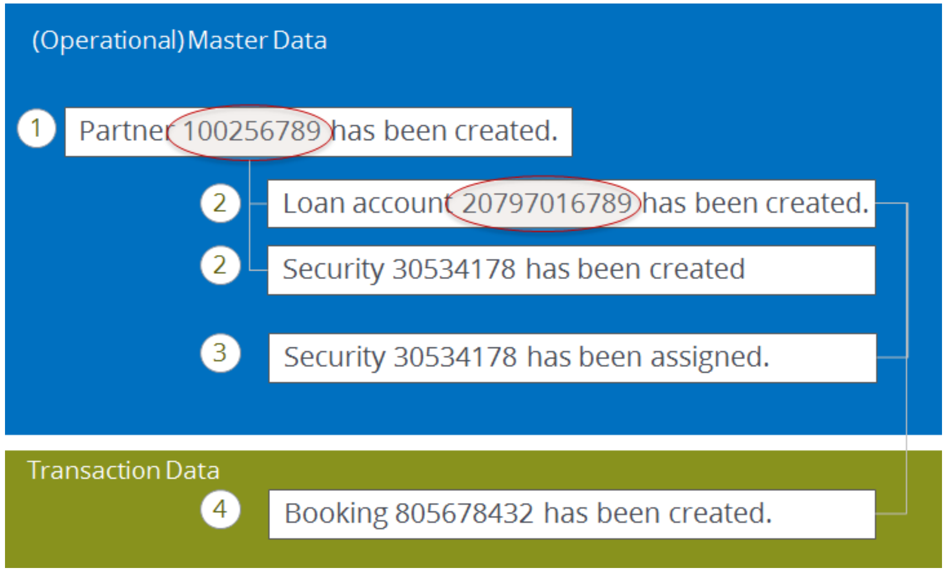
Graphic: Use API access for the automated generation/ administration of synthetic test data whenever possible.

How to deal with dynamic IDs

The creation of test data objects needs to follow their hierarchy (1 … *business partner*, 2 … *account*, etc.). The app will dynamically assign IDs to newly created objects (e.g. ID of a new *business partner*). Subsequently created objects (e.g. *account*) will need to refer to the dynamic IDs of their parent objects (*business partner*).

As a consequence, your data generator must not refer to hard-coded IDs, but needs to select the appropriate dynamic IDs at run-time instead.

Tosca's Test Data Management has built-in functionality to register and administer dynamic IDs.


(Operational) Master Data
1
Partner 100256789 has been created.
2
Loan account 20797016789 has been created.
2
Security 30534178 has been created
3
Security 30534178 has been assigned.
Transaction Data
4
Booking 805678432 has been created.

Graphic: The creation of test data objects needs to follow their hierarchy. Dynamic IDs of parents need to be referenced by their depending objects.

3

TRICENTIS

Connect multi-step scenarios with a logical time axis

While the creation of test data objects is rather straight forward, those objects might still require **end-of-day processing** to be ready for further treatment. E.g. Swiss banks will do in-depth batch-checks on new *business partners*, before they can become an *account holder*.

Even more complex is the task of bringing test data objects into the desired state. E.g. an *account* may require a certain *bookings* history to trigger specific calculations of *interest rates*.

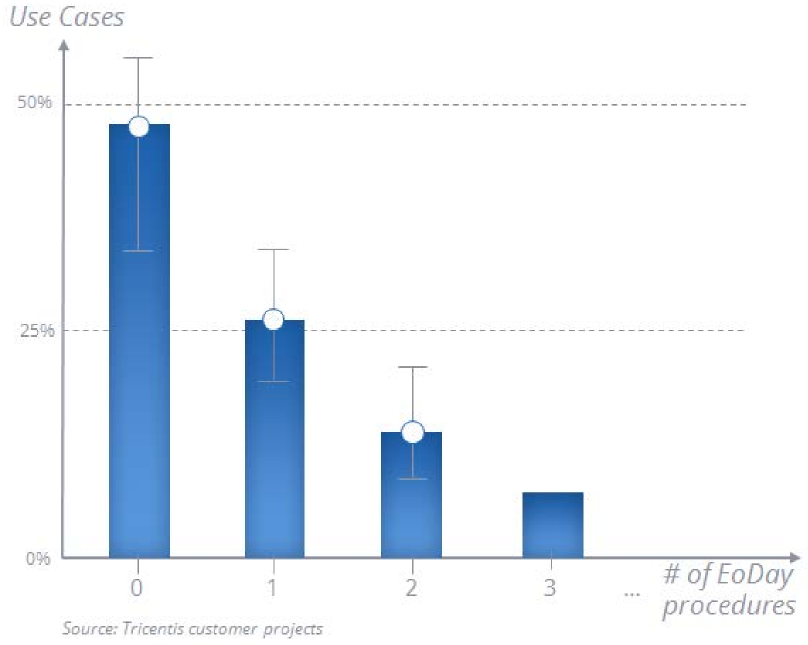
According to Tricentis' customer projects, more than 50% of core-banking use-cases require one or more previously run **end-of-day procedures**. The vast majority can be run on the same **business day**, but some require end-of-day or even end-of-month routines to happen on very specific dates. All in all, to comprehensively test a complex bank you will need 20+ logical business days to run through.

Setting-up a powerful test data management system requires **implementing a logical time axis**, holding the required **business days**. The days need to be dynamic as well: as time goes by, the *15th of the month* will be the *15th of May*, the *15th of June*, etc.. Your automated test data generator needs to link to those dynamic dates.

Customers benefit from **Tosca's built-in concepts and the domain expertise of Tricentis** and its partners.

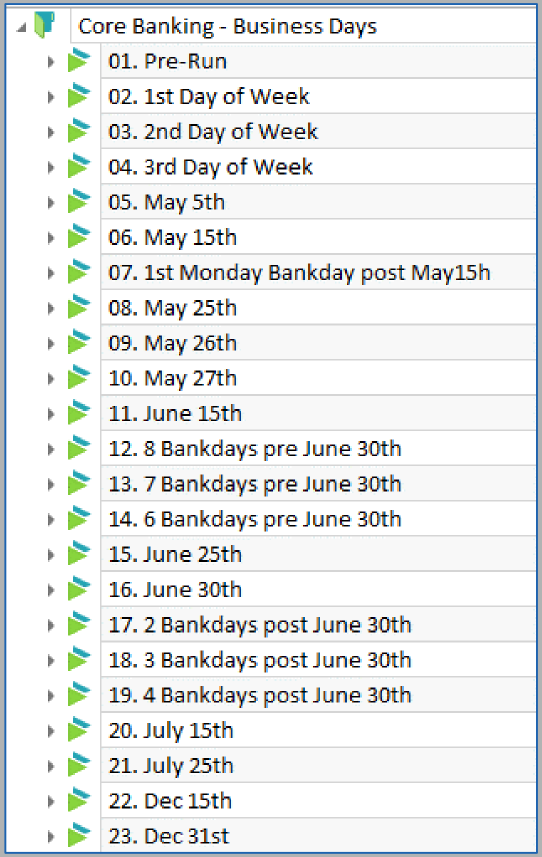
50%

of core banking use-cases require one or more previously run end-of-day procedures.



“After we had designed test cases and their required test data basis together with Tricentis, we derived 25 logical business days. Today our tests are run on a rigid battle-plan. Through dynamic links to business dates, maintenance in ***Tosca’s Test Data Management*** *is avoided successfully.”*

QA director, major retail bank



Core Banking - Business Days
- 01. Pre-Run
- 02. 1st Day of Week
- 03. 2nd Day of Week
- 04. 3rd Day of Week
- 05. May 5th
- 06. May 15th
- 07. 1st Monday Bankday post May15h
- 08. May 25th
- 09. May 26th
- 10. May 27th
- 11. June 15th
- 12. 8 Bankdays pre June 30th
- 13. 7 Bankdays pre June 30th
- 14. 6 Bankdays pre June 30th
- 15. June 25th
- 16. June 30th
- 17. 2 Bankdays post June 30th
- 18. 3 Bankdays post June 30th
- 19. 4 Bankdays post June 30th
- 20. July 15th
- 21. July 25th
- 22. Dec 15th
- 23. Dec 31st

Graphic: A comprehensive test of a core banking app requires 20+ logical business days.

4

Complement remaining leaks through extracts of production data

Synthetic test data generation sometimes fails when data objects with a long lasting history are required. E.g.: It might be difficult or even impossible to provide a 40-years-life insurance contract that has been signed 25 years ago. Such “old” data constellation will be part of production data however.

Deeper investigations reveal that this problem is rather small. On average, less than 2% of the test cases require data that cannot be provided synthetically. In addition, the criticality of use-cases declines with the age of data objects: If a certain use-case cannot be performed for a couple of days, this is in general less critical for a 25 year old contract than a very new one.

The use of synthetic and production data is complementary: synthetic data can be added to production data and vice versa. Ideally customers use extractors/loaders to identify and load data objects that cannot be provided synthetically. Since the dimension of the problem is very limited, even direct SQL insertion is a valid option and anonymization is less of importance.

Through its data-base engine, Tosca provides a powerful toolset to set up extractors/loaders of production data.

<2%

On average, less than 2% of test cases require data that *cannot* be provided synthetically.

5

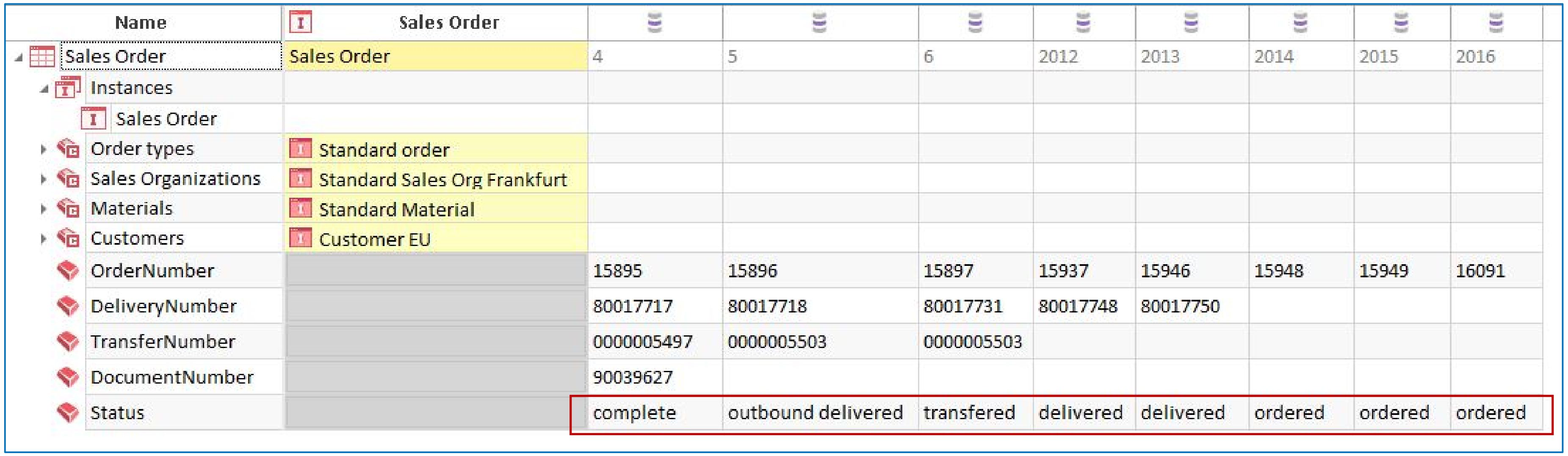
TRICENTIS

Obtain stateful test data management

The power of synthetic test data is a well known fact in the testing industry. However, the implementation success is rather low. In addition to unresolved challenges with dynamic IDs and date/times, customers face ongoing changes that testing creates in their test databases.

Imagine you would grant a *loan* to a certain *business partner*. The *loan* can be approved, since the *business partner* has a good customer rating. Through granting the loan, the rating of the *business partner* is lowered, disqualifying him from getting the loan repeatedly. From the perspective of the use-case "*Create loan account*", the *business partner* has been consumed.

Or, even more challenging, the *order-to-cash* process in your SAP environment needs to be tested comprehensively. The *sales order* will run through a series of different states, which will be required as base-states for the subsequent test.



Name	Sales Order								
Sales Order	Sales Order	4	5	6	2012	2013	2014	2015	2016
Instances									
Sales Order									
Order types	Standard order								
Sales Organizations	Standard Sales Org Frankfurt								
Materials	Standard Material								
Customers	Customer EU								
OrderNumber		15895	15896	15897	15937	15946	15948	15949	16091
DeliveryNumber		80017717	80017718	80017731	80017748	80017750			
TransferNumber		0000005497	0000005503	0000005503					
DocumentNumber		90039627							
Status		complete	outbound delivered	transfered	delivered	delivered	ordered	ordered	ordered

Graphic: List of standard sales orders with different states, tracked by Tosca Testsuite™.

Stay in touch

Tricentis GmbH
Saturn Tower
Leonard Bernstein Straße 10
1220 Vienna
Austria