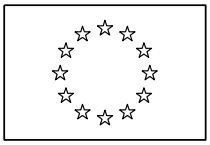


EUROPEAN COMMISSION
DIRECTORATE-GENERAL
INFORMATICS
Infrastructure Directorate
**Technical solutions and office systems**

DIGIT

# Digit Test Centre

# Guidelines and procedures for Information Systems performance testing

| | |
|---|---|
| Date: | 29/11/2010 |
| Version: | 1.000 |
| Authors: | Zaarour Alexandre |
| Revised by: | Babiano Gabriel, Pascal Brahy |
| Approved by: | Babiano Gabriel |
| Public: | Project Leaders, Project Managers, Data Centre, Test Centre |
| Reference Number: | ref |

# Document History

| Version | Date | Comment | Modified Pages |
|---|---|---|---|
| 1.000 | 29/11/2010 | Document created by Pascal Brahy | |
| | | | |

Contact: Alexandre Zaarour, Telephone:(352) 43 01-34788, alexandre.zaarour@ext.cec.eu.int

## 1. INTRODUCTION

Quality is becoming more and more important with the higher visibility of applications and critical role of the Information Systems.

Testing gives an overview on the quality of an application. To achieve this, the testing activity tries to find as many problems as possible in an Information System Thus, it may be concluded that the purpose of testing is to find problems in a system (and the purpose of finding problems is to get them fixed).

A formal definition by the IEEE follows: "*testing is the process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component*"1.

To cover the different aspects of a system, different levels of testing (unit, integration, system, acceptance, preproduction and production) and different types of test (functional, non-regression, technical, performance, exploitability …) can be conducted.

The next section will introduce notions about testing activities.

## 2. OBJECTIVES OF THIS DOCUMENT

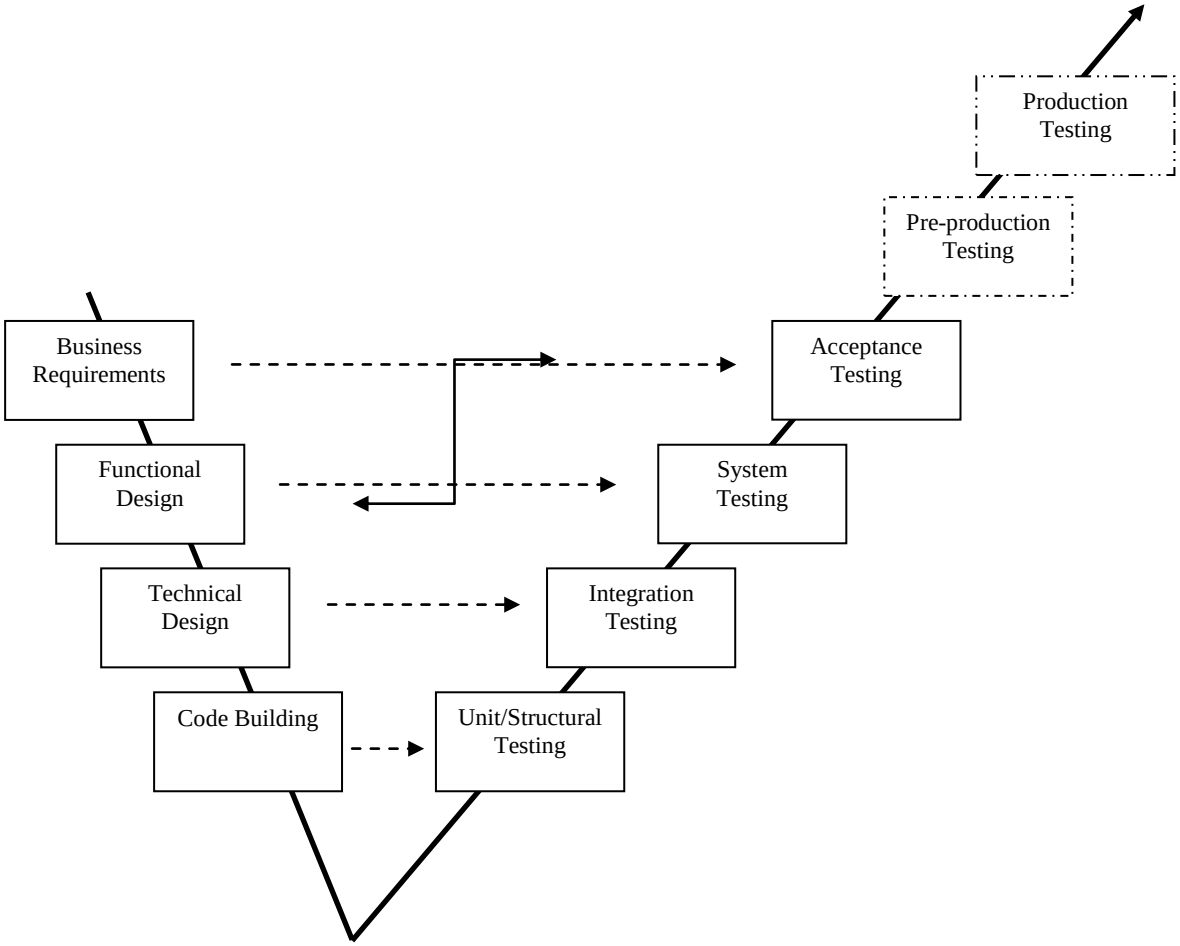
You are reading this document because you requested the test of an Information System you plan to centrally deploy or because it has been requested by the production Change Advisory Board (CAB).

This document is a strongly recommended input in order to understand how to efficiently request the service of the DIGIT TESTCENTRE (hence to efficiently plan your Information System acceptance tests) considering the following aspects:

- General testing information
- The testing scope
- The procedures, roles and responsibilities
- The mandatory inputs
- The testing environment
- The testing strategy and workflow
- The testing deliverables

## 3. GENERAL TESTING INFORMATION

Testing is an activity which finds its place during every step of the application life-cycle. The "V" model is often used to depict this concept.




The above schema illustrates the relations between parts of the application life-cycle and the corresponding category of testing activity.

Each of these categories can consist in different types of test (functional, structural, alpha, beta, regression, performance…) at different levels (unit, integration, system…).

In this document, the tests are presented in two main categories:

- The white box testing where access to the internals of the System under test is open. (Source code is available and required) It mainly happens during the application development cycle.
- The black box testing where access to the internals of the System under test is considered closed. (Availability of the application source code is optional). It usually happens in pre-production or in production.

### 3.1. Testing activities during the development phase (White-box)

| Testing activities | Description |
|---|---|
| Unit testing | The unit testing practically consist in the creation of tests drivers calling the original application code using known interfaces and specific parameters to assess that an expected value is still returned by the called part of the application. This code is invoked by a unit testing application; either a specific component of an IDE of a standalone specialized application, or a custom developed application to verify the functionality of a single unit of code. |
| Structural testing | Testing that takes into account the internal mechanism of a system or component. Types include branch testing, path testing, statement testing. |
| Integration testing | This test is tightly related to the way modern applications are built: assembling components or services, respectively in component based and service based application developments.<br><br>The integration tests are done to verify if two or more related part of an application are achieving the functionality without introducing side effects or instability.<br><br>Depending on the size of the application, there may be a specific software integration staff responsible as well for the integration tests. |
| System testing | Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. |
| Acceptance testing | All testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.<br><br>The acceptance testing also called user acceptance testing or beta-testing and is usually conducted by a relevant selection of the future users of the application. They test the software according to the specified business requirements and report any discovered abnormalities.<br><br>The coordination of the acceptance test is done by the test manager but the most important part of the work is done by the beta testers. |

## 3.2. Testing activities during the pre-production phase (back-box)

Pre-production tests take great importance since they are the first opportunity where application behaviour can be measured in a production-like environment. This is particularly true for projects being deployed at the Data Centre.

Pre-production tests are especially important for intranet/internet applications. Due to their nature (they have potentially a huge number of users and they have a distributed architecture where several components are used), it is necessary to study their behaviour under peak loads. Questions like "how many users will the application cope with?"; "what is the impact of the number of connected users on the performance?"; "which is the component that will become the first bottleneck with a high load?"; etc need to be answered during the pre-production phase.

While some authors include non-regression testing as part of the pre-production tests as they are carried out immediately after the acceptance tests, which include all sorts of tests (functional, regression, etc), the rest of the document will focus on the performance aspects of the pre-production tests.

The pre-production testing is a suite of tests conducted on the application in a production-like environment with the following goals:

- To check if the application meets predefined criterions like business transaction response times
- To make sure the application will behave correctly without risks for the surrounding environments
- To assess the application stability over time
- To determine the resource needed to host the application

This suite of tests can result in the *qualification* or non-qualification of the Information System for production.

The following types of tests are conducted to respectively assess the goals listed above.

| Testing activities | Description |
|---|---|
| Load Testing | The load testing is the activity of running defined use-case of the application, increasing number of users following a defined scenario, in an environment as similar as possible to the real production, while controlling defined metrics, until defined application/infrastructure limits are reached.<br><br>It also collects the business transactions response time to compare them with the specifications. |
| Stress Testing | The stress testing is the activity of running defined use-cases of the application, using a large number of users (usually more than expected under normal load) following a defined scenario, in an environment as similar as possible to the real production, while controlling defined metrics on the system/infrastructure. |
| Endurance Testing | The endurance testing is a load testing during a defined extended period of time in order to check the application and infrastructure stability (no memory leaks, availability of resources, etc) under load conditions. |
| Capacity Planning | The capacity planning is the activity of determining the resource that will be needed in production to host the Information System in meeting the specified acceptance criterion (transaction response time, maximum |

| | |
|---|---|
| | concurrent users…) |
| Bottleneck detection | The bottleneck detection is the process of finding the slowest part of the application using specialized introspection tools. Depending on the technology used to develop the application, the output of this phase can range from general information (ex: which tier is impacting the performance) to very detailed findings (ex: what SQL statement, what EJB is responsible). |
| Pre-production Tuning | Tuning first happens during the pre-production phase. It consists in the monitoring of the application while it is under test with the goal of finding the best configuration parameters (allocated memory or CPUs, Databases index…) for the production deployment while meeting the tests specifications. It also helps in the capacity planning. |

### 3.3. Testing activities during the production phase (black-box)

The production testing is the testing of the application in its production environment at regular defined intervals to check that the constraints of the service-level agreement are constantly met and to report them to the operators if it is not the case.

| Testing activities | Description |
|---|---|
| Monitoring | One of the best practices in monitoring is to reuse the tests scripts and scenario defined during the development and pre-production phases and to execute them with a specific tool, specifying the frequency of their execution and the metrics to be collected, monitoring the "heartbeat" of the application.<br><br>Monitoring is mainly used to assess the Service Level Agreements. |
| Production Tuning | Tuning can also happen during the production phase. It consists in the monitoring of the application with the goal of changing the configuration parameters (allocated memory or CPUs, Databases index…) in order to reach better performance or to release unused resources. |

# 4. TEST STRATEGY

## 4.1. Testing scope

The DIGIT Test Centre focuses on the test activities taking place after the software has been developed, the code is considered optimized and is functionally correct: the pre-production test category.

These pre-production tests are critical for the following reasons:

| |
|---|
| To guarantee the stability of the information system |
| To assess the performance (Response times) |
| To evaluation the necessary resources required to host the application |
| To determine the most efficient software configuration |
| To assess the large load / number of users supported by the Information system |
| To answer to specific performance problems encountered by an application in production |

The DIGIT Test Centre currently delivers the following pre-production testing services:

- Assistance for the definition of the pre-production tests specifications

  If requested by the DG, the Test Centre can help for the specification of the tests. Its added-value in this context is of course not in the business part of the specifications but to help in understanding how to efficiently request the service in a commonly understood and agreed way.

- Assessment and validation of the pre-production tests specifications

  The Tester in charge will first make sure that the returned specification is adequate and realistic. She/He will optionally contact the DG to propose modifications when applicable.

- Test scripting
- Performance (load/stress/endurance) testing
- Capacity planning
- Pre-production Tuning
- Bottleneck detection
- Test reporting

## 4.2. Out of scope

The DIGIT Test Centre is not involved and responsible of:

- Installing the pre-production environment
- Déploying applications in the pre-production environment
- Configuring application in the pre-production environment
- Writing pre-production test specifications
- Correcting code related to performance issues

- Deciding of the Information System go live for production

## 4.3. Testing methodology (workflow)

The purpose of the testing methodology is to describe the chronology and strategy of the potential actions to be taken during the Information System test cycle.

Some of the steps are executed all the time while others are optional, depending on the System under test. Referring to types of tests listed in the section "Testing scope", the methodology is described in the following sub-chapters.

### *4.3.1. General strategy*

As a general rule during the test campaign, only the same version of an Information System is tested. This is to avoid confusion in the process and ending up with incomparable results.

All the components involved in the testing are also monitored in order to make sure they are not a bottleneck which would prevent usable and correct results to be delivered. This monitoring is also useful to determine the impact of the different tests on the resources available on the hosting environment.

At this stage, we consider that the system under test is available and functionally verified.

The methodology aims at returning as soon as possible valuable information on the System under Tests to the DG. This allows an impact analysis on the development cycle and early correction of major issues before going to production.

### *4.3.2. « Aggressive » load test*

After the implementation of the easiest test script, an aggressive load test is executed.

An aggressive load test is the execution of the test script with the maximum number of declared concurrent users in peak situation. (cf. Test specification) The concurrent users are all started at the same time and the different step of the scripts are executed without "think time" (wait time) between the different business transactions.

This step is critical to find the first issues commonly met in to most of the tests and to correct them before going into more complicated phases. Typical issues are either at the hosting environment level or specific to the application configuration.

Typical issues RELATED TO the hosting environment:

- Not enough memory configured
- Not enough Oracle process on the DB side
- Not enough Oracle connections on the Application Server side
- Concurrency errors due to bugged server version
- …

Typical issues RELATED TO the application configuration:

- Not enough identified users created
- Wrong users information (password, roles, …)
- Predefined business conditions not present, not well initialised or not well configured
- Inappropriate application and/or framework parameters configuration

This test also aims to come out application errors like ORA-xxx (Unique Constraint violated), BEA-xxx ones (like java objects that doesn't implement Serializable …).

This test also allows to quickly determine if the testing and hosting environment are sufficiently sized for the target application. An initial tuning might be done at this stage.

### *4.3.3. First load test*

Once the aggressive load test is finished and the first issues addressed, the first test scenario is executed as detailed in the test specification.

Please refer to the scenario definition and execution chapters to find out what types of combinations are possible.

The outcome of this test is a first result to be reported to the DG via an intermediate report. It will give a first insight on the application and conditionally initiate bottleneck detection if any are identified.

### *4.3.4. Bottleneck detection*

Based on the first load test and monitors configured, potential bottlenecks may be identified. In this stage we will identify specific elements in the code that either slows the application down or cause a high resources consumption like Java heap, machine CPU, etc.

This phase can happen at different time during the test cycle as a reactive investigation to explain slow path of an application with a maximum of detail.

It is currently only possible for applications developed on the J2EE platform / Oracle database and is realized using introspection tools. It requires specific additional components to be deployed along with the application. While these additional components have a global constant impact on the application performance, this is not important as the goal at this stage is to find and explain bottlenecks, not to determine the real performance of the application.

Specialized introspection tools are used for the following common issues:

- J2ee layers

The tools allow the decomposition of Business transactions in Http server requests with large number of statistics (Elapsed time, CPU,) and then to drill down into the slowest server requests at a finer level of detail.

- Statistics about Http Sessions

When high Java heap memory consumption is detected, specific tools are used to display statistics about http sessions in the JVM and then objects remained in the heaviest sessions.

- Java heap memory

Some tools will help us to find memory leaks and reduce memory consumption.

- SQL statements

Oracle introspection tools that are already installed in the hosting machine will be used to find poorly tuned Sql statements, lack of indexes / Full table scan, Sqls executed a high number of times, latches…

Our concern here is to detect oracle elements that cause either high business transaction response times or high CPU resources consumption on the database machine.

However performance problems caused by poorly tuned oracle parameters will not be taken in charge by the TestCentre service.

### *4.3.5. Full load test*

The full load test is the execution of all the qualified test scripts as described in the test specifications.

During their execution, different parameters of the hosting environment or specific to the application might need to be adapted to meet the requirements. They will be collected and reported for the capacity planning.

It is a substantial suite of tests resulting in an exhaustive report detailing the outcomes of all the business transactions and their impact on the hosting environment. (The Test Report is described in a separate document)

### *4.3.6. Stress test*

The stress test is the optional execution of the load test in an aggressive way either by:

- Executing more users than specified
- Pacing the execution more aggressively then specified

The purpose of this stage is to further assess the stability of the application under heavy use and also to have an approximation of the resources needed if the usage of the application goes over expectations.

Once again, the different parameters of the hosting environment or specific to the application might need to be adapted and will be collected and reported for the capacity planning.

### *4.3.7. Stability test*

The stability test is the execution of a load test executed during a prolongated period in order to assess the stability over time, the transaction response times and resssources consumption.

### *4.3.8. Capacity planning*

The capacity planning is not specifically a test activity but is the collection of the "best parameters" set/adapted during the load and the stress tests.

Reporting them will help the production team to anticipate the needs to host the applications and to configure optimal resource allocation to meet the usage specifications.

## 4.4. Closure of tests: Main criteria

The performance tests outcome is taken into account as important criteria of decision during the cab meeting to decide whether the information system will receive a go live for the production environment.

The Testcentre service will execute the load tests as described in the test methodology chapter and provide results to the change Coordinator and the concerned hosting teams after the test campaign is consider as complete.

The test campaign can be considered as complete when:

- All the scenario have been executed as specified by the DGs (Full Load test)
- All the scenario results match the expected ones in term of response times
- The system under test is considered as stable (Stability test)
- The resources consumed by the application are juged acceptable by the Datacenter teams.

The stability of a system mainly include notions like response times stability, error rates inferior to 5% and no system crash after a long period of load.

The hosting teams, concerned by the technologies used by the system under test, primarily validate the resources consumption (CPU, memory). Based on their feedbacks, new load tests can be executed.

### 4.5. Risks / Potential prerequisites

This section list the potential prerequisites that must be taken into account by the DGs during the installation and configuration of the information system to be stress tested.

It also describes, for each prerequisite, the actions to be executed in order to avoid these risks.

| *Risk* | *Description / Impacts* | *Actions* |
|---|---|---|
| Protocol /Technology not supported | Some technologies like Flex3 are not supported by Loadrunner and can not be scripted. | 1. The section "***testable types of application and protocols***" indicate which protocols are supported by the DIGIT TestCentre<br>2. Ask the DIGIT TESTCENTRE to do a ***test feasibility***. |
| Application bugged | An application that contains too many functional bugs will prevent testers to script the test cases. | 1. The DGs must provide an application release for which the code is considered optimized and is functionally correct.<br>2. Once you have a confirmation that the application is installed, you have to validate this installation against the specified tests to make sure everything is fully functional |
| Bad configuration of the application | Application badly configured will result in preventing testers to script the test cases. | Once you have a confirmation that the application is installed, you have to validate this installation against the specified tests to make sure everything is fully functional |
| Database & Test data | Database that are not populated with realistic data or that haven't enough data set will produce unrealistic performances dashboard. | 1. DGs must make sure that the database installed and configured for the test is in a consistent and fully functional state considering the requirement of the tests specification.<br>2. Any other necessary assets (flat files, XML files) must also be verified. |
| Database & Initial Dump | If the Information System specified tests implies both reading and writing to the database, the increase of database size will most probably impact applicative performances. | 1. It's necessary to restore a "fresh" database before each test.<br>2. The TC in collaboration with the DC has elaborated ***an easy accessible and executable procedure*** to restore the database to its initial state.<br>3. It is advised that the requesting DG |

| | | |
|---|---|---|
| | | keeps on there side a ***backup of the initial database*** in case of restore point is lost due to any technical reason. |
| External resources | When the application contains links to resources (images, css, etc) hosted on the production, the stress test may impact those external servers. | In order to prevent perturbations on the production, DGs must archive all the applicative resources on the same WAR or EAR. |
| Users & Roles | Most of the Information Systems test will require the usage of separate application users with different roles. If these ones are not configured, some business functions will not be accessible. | 1. DGs must make sure that the ***appropriate users and roles*** are ***created*** and usable to implement the specified tests. |
| ECAS configuration | More and more application developed by the EC use the ECAS authentication system. This system required some specific application side configurations. | 1. DGs must configure the Ecas inside the application as specified hereunder.<br>2. The Ecas usernames are like in production. The DGs must use those ones. |

**How to configure the application for ECAS*?***

The configuration of the ECAS client inside the applications is part of the development and there exists documentation of how to do it: http://www.cc.cec/wikis/display/IAM/ECAS

The ECAS binaries are put inside the classpath of the servers but the configuration files are part of the application itself.

Here is the information about ECAS/LDAP:

| Ecas/Ldap | Machine | Port | Usernames | Password |
|---|---|---|---|---|
| ECAS | Ecasl.cc.cec.eu.int | 7002 | Like in production + uid00000 à uid29999 | Str3ssm3 |
| LDAP | cedstrs1.cc.cec.eu.int | 10389 | Like in production + uid00000 à uid29999 | Str3ssm3 |

The Properties to set for ECAS are:

.serviceUrl=http://[your appserver]:[port]/[your login page]

.loginUrl=https://ecasl.cc.cec.eu.int:7002/cas/login

.validateUrl=https://ecasl.cc.cec.eu.int:7002/cas/laxValidate

The parameter "acceptStrength" should be set to "BASIC".

## 5. TEST ENVIRONMENT

### 5.1. Human resources

The currently available human resources are one Service Manager and six full time Testers provided through the STIS-LOT4 framework contract.

### 5.2. Software resources

| Testing activity | Tool(s) used | SW Vendors |
|---|---|---|
| Assistance for the definition of the pre-production tests specifications | none | |
| Assessment and validation of the pre-production tests specifications | none | |
| Test scripting | Vugen | Hp Mercury Interactive |
| Load testing | LoadRunner | Hp Mercury Interactive |
| Stress testing | LoadRunner | Hp Mercury Interactive |
| Endurance testing | LoadRunner | Hp Mercury Interactive |
| Capacity planning | LoadRunner | Hp Mercury Interactive |
| Bottleneck detection | LoadRunner J2EE diagnostic, Opensource tools. | Hp Mercury Interactive |
| Resources Monitoring | Sitescope | Hp Mercury Interactive |
| Tuning | LoadRunner, Oracle tools | Hp Mercury Interactive |
| Test Reporting | LoadRunner Analysis, Word+Excel | Hp Mercury Interactive |

### 5.3. Hardware resources

The pre-production environment is designed such to be as closed as the production like and is composed of the following machines:

| *Type* | *Name* | *Machine Name* | *System* | *Number of CPU* | *Memory Size* |
|---|---|---|---|---|---|
| Loadrunner Injectors | s-net1lux-strt3.net1.cec.eu.int | NA | MS 2003 | 4 CPU Intel Xeon 3.66GHz | 8 GB |
| | s-net1lux-strt4.net1.cec.eu.int | NA | MS 2003 | 4 CPU Intel Xeon 3.66GHz | 8 GB |
| | s-net1lux-strt5.net1.cec.eu.int | NA | MS 2003 | 8 CPU Intel Xeon 3.20GHz | 8 GB |
| | s-net1lux-strt6.net1.cec.eu.int | NA | MS 2003 | 8 CPU Intel Xeon 3.20GHz | 8 GB |
| | s-net1lux-strt7.net1.cec.eu.int | NA | MS 2003 | 8 CPU Intel Xeon 2GHz | 8 GB |
| | s-net1lux-strt8.net1.cec.eu.int | NA | MS 2003 | 8 CPU Intel Xeon 2GHz | 8 GB |
| | s-net1bru-strt7.net1.cec.eu.int | NA | MS 2003 | 8 CPU Intel Xeon 2GHz | 8 GB |
| | s-net1bru-strt8.net1.cec.eu.int | NA | MS 2003 | 8 CPU Intel Xeon 3.20GHz | 8 GB |
| ECAS | Elk | NA | Sun | 8 CPU sun4u Sun Fire 880 | 32 GB |
| Oracle | Oraload1 | mike1 | Sun | 32 CPU sun4u Sun-Fire-15000 | 12 GB |
| | Oraload2 | NA | Sun | 24 CPU sun4u Sun-Fire-15000 | 98304 MB |
| | Oraload3 | NA | Sun | 24 CPU sun4u Sun-Fire-15000 | 98304 MB |
| | Oraload4 | NA | Sun | 16 CPU sun4u Sun-Fire-15000 | 65536 MB |
| Weblogic | Wlsload3 | chipmunk | Sun | 64 CPU SPARC-Enterprise-T5220 | 32 GB |
| | Wlsload4 | pipit | Sun | 32 CPU SPARC-Enterprise-T5220 | 16 GB |
| | Wlsload5 | shrike | Sun | 32 CPU SPARC-Enterprise-T5240 | 16 GB |
| | Wlsload6 | shrike | Sun | 32 CPU SPARC-Enterprise-T5240 | 16 GB |
| Coldfusion | Grouse | NA | Sun | 2 CPU Sun Fire V210 | 8 GB |
| | Cf8glos1 | tamarin | Sun | 64 CPU (zone:64) SPARC-Enterprise-T5220 | 32 GB |
| | Cf8glos2 | reeding | Sun | 128 CPU (zone:40) SPARC-Enterprise-T5240 | 32 GB |
| BO | Webiload1/ oryx | NA | Sun | 16 CPU Sun4u-Fire-V890 | 65 GB |
| | Bodiload1 | NA | Sun | 16 CPU Sun4u-Fire-V890 | 65 GB |
| IAS | Oasload1 | chipmunk | Sun | 64 CPU SPARC-Enterprise-T5220 | 32 GB |

| | | | | |
|---|---|---|---|---|
| Oasload2 | chipmunk | Sun | 64 CPU SPARC-Enterprise-T5220 | 32 GB |

## 6. CONCEPTION OF THE TESTS

The generation of load is achieved using specialised tools that simulate user interactions with the application. These tools are known as Load Test Tools and they require a considerable economic investment and certain specialised knowledge to use their power to simulate meaningful usage scenarios.

All these considerations lead to the conclusion that a common independent body, a Load Test Centre (referred to as Test Centre), shared by all projects is a cost-effective answer to the need of executing pre-production tests.

This section will depict the 4 main steps of a test campaign:

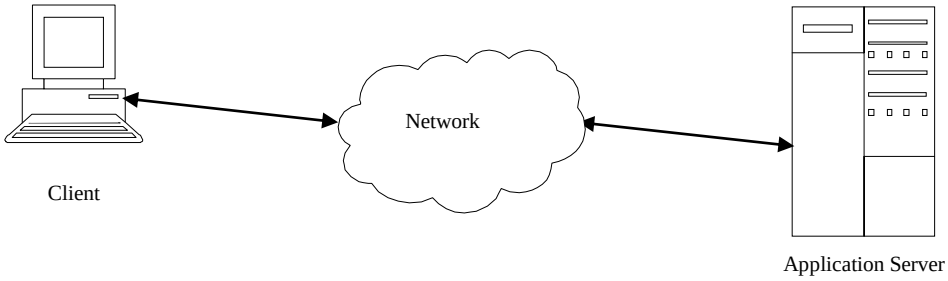
- Recording the test scripts
- Running the test scenario
- Collecting the metrics
- Analysing and reporting on the test data

### 6.1. Recording the test script

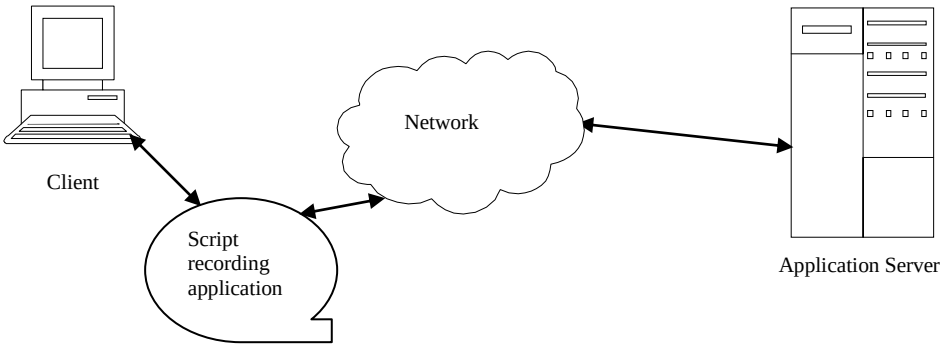
Two different types of script are considered in this section: the test cases and test scenario.

#### *6.1.1. Technical principle of scripts recording*

The following schema is a generalized representation of the original execution environment of the application to be tested.




This second schema illustrates how the script recorder is inserted in the application chain to act as a pass-through while recoding every exchange between the tiers of the application on defined protocol(s). It is important to emphasize that the client application transparently continues work like before the presence of the recorder.




Most testing tools and suites support the recording of a wide range of protocols and are capable of recording multiple protocols at the same time.

### *6.1.2. Defining the test case*

A test case is a sequence of business transactions which are composed of atomic actions. In other words, a test case is an ordered sequence of operations with the goal of achieving a business process.

Each exchange between the client and the server will be surrounded by a marker that names the action and measures the time between the http request and http response.

Specific functions are inserted before the actions to verify whether or not the page you received is the desired one. This helps to calculate the error rate by action during the running phase.

Think times are included before the actions as described by the DGs in the test specification document.

Some actions that don't imply exchange with the server need to set data using text field, dropdown list, check box, radio button, etc. To test more realistically and avoid server side caching, sets of data must be used in the script. This is done by entering data as parameters which can possibly be taken out of different data sources.

*Internal data:*

Data that is generated internally by the scripts:

- Date/Time
- Group Name
- Host Name
- Iteration Number
- Random Number
- Unique Number
- Virtual users ID (Vuser ID)
- Parameters returned by the application

*Data files:*

Data are contained in an external file. Each user can pick a value up sequentially, randomly or keep the same value for all the iterations.

*User-defined function based data*

Data generated using an internal or external function.

Considering all the here above remarks, the test requestor must clearly detail the actions to be executed, their order and the data to be used in the test specification document.

### *6.1.3. Defining the test scenario*

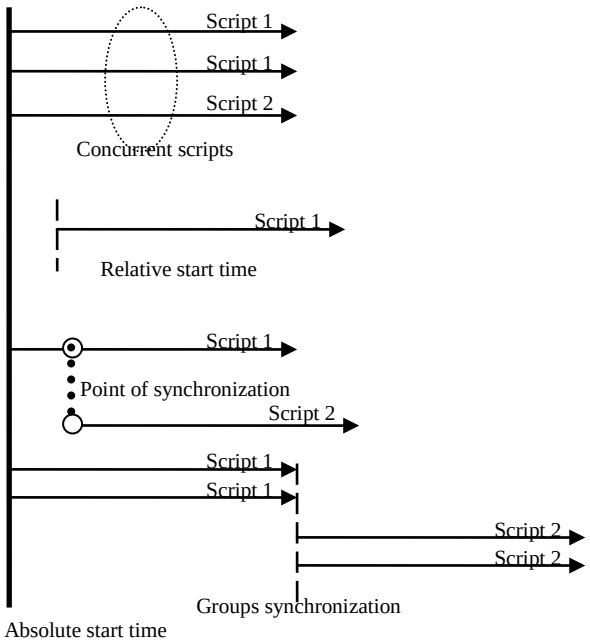
The scenario is the way the test scripts are replayed. It is defined depending on the way the application has to be tested and the goals to be reached.

It is possible to combine different test scripts to model the real-life situations. This is useful to measure the impact of functionally different tasks accessing the same resources.

The scheduling of the different tasks must be specified and agreed in advance. There are different aspects which need to be detailed:

- Delay in starting a scenario relative to absolute start time
- Delay in starting a scenario relative to another scenario start time
- Dependencies between scenarios
- Concurrency
- Synchronisations

An easy way to represent the scenario and its timing is to represent a timeline with the different scripts involved.




It's possible to shift the users when launching (Ramp up) or stopping them (Ramp down).

Think times may be replayed depending on the test type: Aggressive load or First load.

In case the first load test shows unrealistic, the time each user waits between two test case is increased. This introduces the Pacing notion.

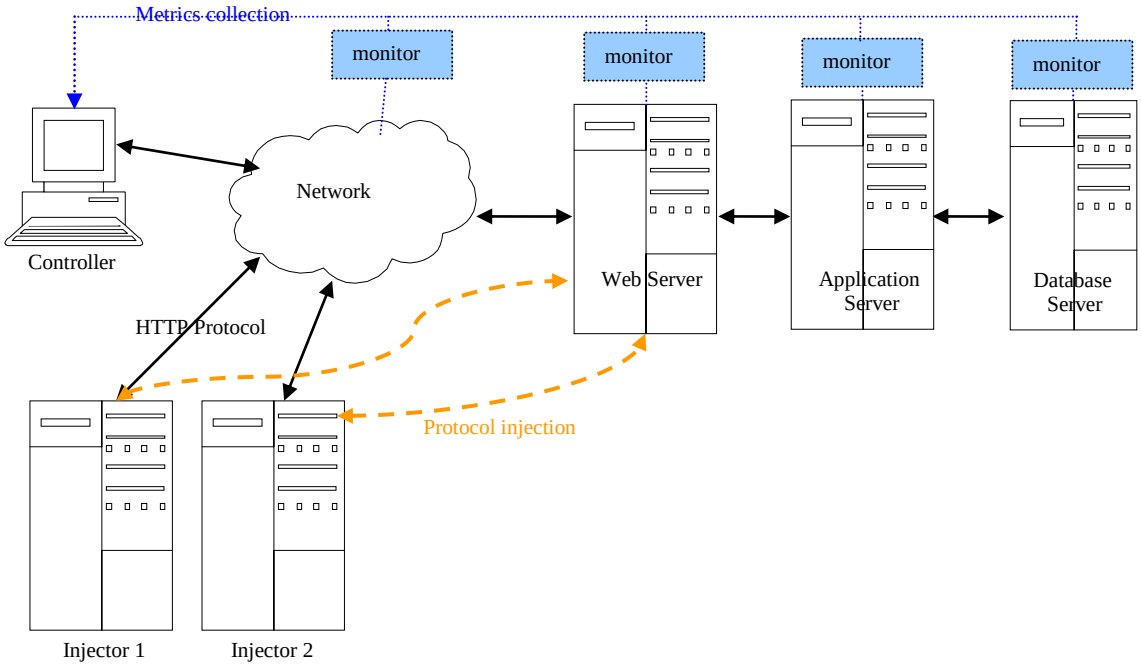
The browser options can also be adapted to be set as the policies applied at the European Commission. For instance, the URL (http) are cached as the URLs (https) are not. The browser cache can also be cleared before each test case iteration.

## 6.2. Running the test scenario

Once the scenario is defined and implemented, the test infrastructure will be able to execute it. To achieve this functionality, a testing suite is typically composed of the following tools:

A monitoring and execution console: the Controller

One or more "injector" machines representing the client




All components can be geographically dispersed. For instance, the injectors might be requested to be dispersed to simulate a multi-site/multi-country access to the applications.

During the execution of the tests, a certain number of metrics is usually gathered about the main the application and its executing environment.

## 6.3. Collecting the metrics

### *6.3.1. Load metrics*

- Hits per second
- Throughtput
- Total transaction per second

- Running users
- Error statistics

### *6.3.2. Application metrics*

- Transaction response time

### *6.3.3. Application execution environment metrics*

- Operating system metrics
- Networks appliance metrics
- Web Server metrics
- Application server metrics
- Database metrics

## 6.4. Analysing and reporting on the test data

At the end of a test session, all the data gathered are either stored in a file system or in a database.

The next phase is the production of a structured, clear and comparable test report.

This work is done after the test session and is of major importance as the analysis and reporting capacities will be critical in finding and explaining the different behaviours experienced during the tests.

It is also a valuable input for the deployment in production.

It is very helpful to have overlaid data from different sources to be able to explain some unexpected or complicated behaviours during the test session. For instance, an abnormally heavy loaded database server might induce an overall slow running of some transactions.

Therefore, a template for reporting the Information Systems tests has been designed. It helps to send the results as soon as possible, with a minimum of efforts in standardized and efficient manner.

Three types of report are sent either during or at the end of the tests:

### *6.4.1. Preliminary report*

The preliminary report is sent as soon as an issue is detected or to inform as soon as possible the Information Systems owner on the capacity of the tested Information System.

Issues fall in the following category:

- Functional errors preventing the implementation of the test cases
- Bugs side effects
- Configuration problems

There is no formal template or format for the preliminary reports.

### *6.4.2. Intermediate reports*

The intermediate reports are sent as soon as a test scenario has been executed or when the result of Bottleneck detection is done. This report is intended to be sent to the DGs, the hosting teams and the Change Coordinator.

There is no formal template or format for the intermediate reports.

### *6.4.3. Final reports*

There is a separate document describing the Information Systems Test report.

## 7. THE TESTABLE TYPE OF APPLICATIONS AND PROTOCOLS

While most applications are implemented using simple and usual technologies, others may have selected less common technologies.

The Test Centre should be able to test all types of application that are supported by the tool it uses. Nevertheless, there are also licenses and other constraints reducing the list.

The following table lists all the "protocols" (c.f. Conception of the tests) supported by the tools, specifies which one are currently supported by the Test Centre and also indicate a weighting of the complexity associated to each protocol.

| Protocol | Supported by LoadRunner | Supported By TC | Complexity factor (1-5) | Minimal duration |
|---|---|---|---|---|
| Web (HTTP/HTTPS) | Y | Y | 1 | 2 weeks |
| Soap/WebServices | Y | Y | 2 | Case-by-case |
| Socket level | Y | Y | 5 | Case-by-case |
| Java | Y | Y | 4-5 | Case-by-case |
| Adobe Flex 2 | Y | Y | 2 | Case-by-case |
| Adobe Flex 3/4 | N | N | / | / |
| JavaScript | Y | N | n.a | n.a |
| RMI | Y | Y | 4-5 | Case-by-case |
| EJB | Y | Y | 4-5 | Case-by-case |
| CORBA | Y | N | n.a. | n.a. |
| COM/DCOM | Y | N | n.a | n.a |
| Oracle 2 tiers | Y | N | n.a. | n.a. |
| SQL Server | Y | N | n.a | n.a. |
| ODBC | Y | N | n.a. | n.a. |
| FTP | Y | N | n.a. | n.a. |
| DNS | Y | N | n.a. | n.a. |
| POP/MAPI | Y | N | n.a. | n.a. |
| IMAP | Y | N | n.a. | n.a. |
| SMTP | Y | N | n.a. | n.a. |
| LDAP | Y | N | n.a. | n.a. |
| Media Player (MMS) | Y | N | n.a. | n.a. |

If the Information System under test implements a mix of these protocols, the following information can also be deduced:

If one of the protocols used by the I.S. is not supported by the Test Centre, a full test of the application might not be achieved. This situation is to be examined in the detail, case by case before returning an indication of feasibility.

The complexity is at best the worst factor of all the used protocols. This is to be confirmed on a case by case basis after the Test Centre has practically assessed the application complexity.

The length of the tests will be proportional to the complexity

# 8. TEST DELIVERABLES

## 8.1. Performance test request form (J2ee Web application)

This chapter describe how to fill the test request form. Every entry is associated to a responsibility among Data Centre (DC), Test Centre (TC) or DG for filling this specific part.

Please refer to the other parts of this document for specific descriptions of terminology or technical concepts.

### *8.1.1. Application information*

| *Entitlement* | *Description* |
|---|---|
| *DG (DG)* | The DG requesting the Information System test |
| Application name (DG) | *The usual name by which the application is known or registered* |
| Expected deadline for the tests (DG) | This is the date when you would like the tests and related deliverables to be delivered |
| Short description (DG) | This is a short description the functionality and usage of the Information System under test |
| URL of the Information System under test (DC) | This is the URL on the stress test environment that will be used by the testers to record the test cases and to execute the load and stress tests.<br><br>It will be communicated by the DC as soon as the application is |
| Number of potential users (DG) | This is the total number of distinct users' known to the application either by counting individual login names, unique IP addresses or any other means |
| Estimated concurrent users in normal situations(DG) | This is the potential quantity of users connected and working concurrently with the application in non-peak situation. Idle users are not taken into account as they are not impacting the load of the system |
| Estimated concurrent users in a peak situations (DG) | This is the potential quantity of users connected and working concurrently with the application in peak usage situation. Idle users are not taken into account as they are not impacting the load on the system |
| Estimated concurrent users in a peak situations | This is the potential quantity of users connected |

| | |
|---|---|
| (DG) | and working concurrently with the application in peak usage situation. Idle users are not taken into account as they are not impacting the load on the system |

### *8.1.2. Contacts information*

Please refer to roles and responsibilities for additional information about the different contacts.

| *Entitlement* | *Description* |
|---|---|
| DG main contact (DG) | She/He is the application owner or a representing person having a global view of the application and the implicated parties (developers, analysts, local system administrators …)<br><br>It usually is the Test Requestor and initial single point of contact. |
| DG technical contact (DG) | The technical contact is an individual knowledgeable both on the functional and architectural aspects of the application to be tested. She/he should also be able to route requests to other contacts having a more detailed view on parts of the application. |
| DG Application Support (DG) | The DG application support is an individual (or functional mailbox) where a request for functional explanation on the usage of the application can be requested. |
| DC Change Coordinator (DC) | She/He is the single point of contact at the Data Centre which has been assigned to the follow-up of the Information System to be hosted. Please refer to Information System Hosting Guidelines for more information on this role. |
| DIGIT TESTCENTRE Contact (TC) | She/He is the single point of contact at the Test Centre, a Tester which has been assigned to the test of the Information System. |

### *8.1.3. Technical information*

| *Entitlement* | *Description* |
|---|---|
| Application Server (DC) | It is the brand of the application server hosting the application under test. Usual types are Weblogic, ColdFusion, IAS … |
| Database Server (DC) | It is the database tier(s) were operational data are stored and extracted. It normally is an Oracle database server hosted on a machine dedicated to the load tests. |
| Web Server (DC) | This is the frontal web server through which the application is accessed (if applicable) |
| Other server (DC) | These are any other server not listed before. |
| Specific protocol(s) between tiers (ex: HTTP, SOAP, RMI…) (DG) | This is the list of protocols used between the different tiers of the application. |
| Other technical information (DG) | For every point, please list any other technical information applicable in the context of the tests. |

### *8.1.4. Business Transactions*

The business transaction represents an ordered sequence of atomic actions corresponding to the usage of specific area or control of an application in order to achieve a goal. An example is detailed at the end of this chapter. One or more Business Transaction(s) will be specified for the construction of the Test Cases.

| *Entitlement* | *Description* |
|---|---|
| Id | This is an id associated with the business transaction which is reused for the specification of the Test Cases. |
| Business Transaction Name | This is a short name representative of the functionality achieved by this business transaction. |
| Atomic Action | An atomic action is a single operation executed using an element of the application user interface of like a button, a list a field… |
| | Each listed Business transaction is composed of one or more ordered atomic actions specifying, if applicable, the following information: |
| | - Expected response time (in seconds) during normal usage (load) situation (in seconds) |
| | - Expected response time (in seconds) |

| | |
|---|---|
| | during heavy usage (load) situation<br>- Usual think time (in seconds) before the activation of the action<br>When such information is not applicable to an atomic action, please leave the corresponding cells blank. |
| Expected average response time in normal situation | This is the expected average response time of a business transaction specified by the application owner, during non-peak situations. |
| Expected average response time in peak situation | This is the expected average response time of a business transaction specified by the application owner, during peak situations. |
| Usual think time | This is the representative think time elapsed before the activation by an end-user of the atomic actions during the realisation of a business transaction. This value can be specified with the following notation:<br>- A constant number. Ex: 1 which means 1 second<br>- An interval. Ex: 1-5 which means a random value between 1 and 5 (included)<br>- A %variation. Ex: 1 +/- 10% which means 0.9 to 1.1 |

**Here is a business transaction example:**

| | | | | | | |
|---|---|---|---|---|---|---|
| BT01 | LOGIN | Enter login | Enter a valid login name in the login field | | | |
| | | Enter password | Enter a valid password name in the password field | | | |
| | | Click login | Click the login button | 5 | 10 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| BT02 | Logout | Click Logout | Click the logout button | | | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| BT03 | Query | Select TAB2 | Select the second TAB by clicking it | 1 | 5 | 1 |
| | | Select criteria | Select a name in the drop-down list | | | 2 |
| | | Click query | Click the query button | 10 | 30 | 1 |

The business transaction “BT01: LOGIN” is composed of 3 atomic actions in the following order:

- Enter login

- Enter password
- Click login

In this particular example, it is only relevant to specify expected response time (normal & peak) for the "click login" as it is the only action implying http exchange with the web server. Consequently, all the other cells have been left empty.

It is also important to stress the order in which the Business Transactions are listed is not important as the sequence of Business Transaction is specified in the Test Cases table.

Additionally, you should not cover all areas of your application (as this is not a functional test) but the one which are representative on its real usage.

### *8.1.5. Test Cases*

The Test Cases are ordered sequences of Business Transactions with the goal of achieving a Business Process.

| Entitlement | Description |
|---|---|
| Id | This is an id associated with the test cases in order to reuse it later on during the Scenario specification. |
| Name | This is a short name representative of the functionality (Business Process) achieved by this Test Case. |
| Description | This is an ordered sequence of Business Transaction separated by a "," corresponding to the execution of a Business Process. |

**Test Case Example**:

| | | |
|---|---|---|
| TC01 | Search employee | BT01, BT03, BT02 |

The Test Case TC01 implementing the Search Employee Business Case is composed of the ordered sequence of BT01, BT03, BT02.

This specifies that to achieve this goal, it is necessary to first execute business transactions BT01 then BT03 and finally BT02.

The same Business Transaction could be reused in the specification of other Test Cases.

Please only specify realistic test cases taking into account the real-life usage of the application.

### *8.1.6. Test Scenario*

A Test Scenario is a combination of Test Cases in order to simulate the real-life usage of an application by several users working concurrently on different Business Processes.

An analysis on the usage of the application can be achieved by survey, web server log analysis or Business Requirements specification analysis. It is a necessary step in order to efficiently determine what potential combinations can exist and which one are relevant and useful in the context of the tests.

| *Entitlement* | *Description* |
|---|---|
| Id | This is an id associated with the Test Scenario. |
| Name | This is a short name representative of the Test Scenario. |
| Description | This is a combination of Test Cases using the formatted notation (Refer to Table Test scenario notation) |

| **Notation** | **Example** | **Explanation** |
|---|---|---|
| ***** | 5*TC01 | TC01 is executed by 5 concurrent users |
| **+** | 1*TC01 + 1*TC02 | TC01 and TC02 are executed concurrently by 1 user |
| **,** | 1*TC01, 1*TC02 | TC02 is executed by one user after TC01 executed by one user is finished |
| ***( )*** | (1*TC01 + 1*TC02), 1*TC03 | TC01 and TC02 are executed concurrently by 1 user then TC03 is executed by 1 user |

Please only specify realistic combinations taking into account the real-life usage of the application.

### *8.1.7. General recommendations*

Here are resumed mistakes commonly done when writing the BT, TC and TS specifications:

- ✓ BT: Don't cover all functionalities. Only describe functionalities that will be accessed in concurrence or those ones that are potentially sensible to concurrency.
- ✓ BT: Avoid writing long Business Transactions. This is generally done when TC and BT are confused.
- ✓ BT: Do not set response times to actions that don't imply http exchanges between the client and the server.
- ✓ BT: Avoid duplicating BT.

| Bad Example | Good Example |
|---|---|
| BT01 Login Role1 | BT01 Login |
| BT01 Login Role2 | The Role is specified in the TC |

- ✓ TC: Avoid doing loops of BTs

| Bad Example |
|---|
| TCx = 10 * (BT01, BT02) , BT03 |

- ✓ Avoid Test scenario that don't have added value in point of view of performances:

| Bad Example |
|---|

| |
|---|
| TS01 = 5*TC01 + 5*TC02 |
| TS02 = 5*TC01 + 5*TC02 + 5*TC03 |
| TS03 = 5*TC01 + 5*TC03 |
| TS01 and TS03 can be deleted |

- ✓ Some TS configurations are not possible due to our load test tool.

| Bad Example |
|---|
| TSx = 5 * (TC01, TC02) , 2* TC03 |
| TSx = TC01, (TC02 + TC03) |

## 8.2. Performance test request form (Bodi)

This chapter describe how to fill the test request form. Every entry is associated to a responsibility among Data Centre (DC), Test Centre (TC) or DG for filling this specific part.

Please refer to the other parts of this document for specific descriptions of terminology or technical concepts.

### *8.2.1. Application information*

| *Entitlement* | *Description* |
|---|---|
| *DG (DG)* | The DG requesting the Information System test |
| Application name (DG) | *The usual name by which the application is known or registered* |
| Expected start of tests (DG) | This is the date when you would like the tests to be started. |
| Expected Production date (DG) | This is the date when you planned to deploy the application. |
| Short description (DG) | This is a short description the functionality and usage of the Information System under test |

### *8.2.2. Contacts information*

Please refer to the 8.1 section of this document.

The name, phone and Email must be entered for each contact information.

### *8.2.3. Technical information*

| *Entitlement* | *Description* |
|---|---|
| Source database(s) | It is the database tier(s) were operational data are extracted. |
| Target database | It is the database tier(s) were operational data are stored. |
| Repository database | It is the database tier(s) were user-created and system objects data are stored. |
| BODI Job server(s) | It is the job server retrieving the job information from the associated repository. |
| BODI console | It is the administration console through which the jobs will be executed. |

The machine name, SID, port, software type, username / Password will be defined for each technical information if relevant.

All the database types quoted are normally Oracle database server.

### *8.2.4. Job Description*

Describe each job to be executed with the following information

- Job name: The exact name of the job
- Job description: A general description of the job
- Execution variable values: Values of the variables to be checked before the job execution.
- Planned start date / time: The planned date / Time at which the job will be executed in the production environment.
- Planned duration: A planned duration time of the job
- Planned frequency: The planned frequency (Every night, day, week, etc) at which the job will be executed in the production environment.

### *8.2.5. Test scenario*

A Test Scenario is a combination of Jobs with the possibility to simulate each one concurrently.

## 8.3. Performance test request form (Web Intelligence)

Application Information and contact Information are described in the "Performance test request form for Bodi" section of this document.

### *8.3.1. Technical information*

| *Entitlement* | *Description* |
|---|---|
| Operational database(s) | It is the database tier(s) were operational data are extracted. |
| Repository database(s) | It is the database tier(s) were user-created and system objects data are stored. |
| WEBI access server | It is the brand of the Web Intelligence server hosting the application under test. |
| WEBI administrator | It is the information to access the Webi administration console. It allows sizing and monitoring resources allocated for the Webi server. |

### *8.3.2. Documents*

The steps to refresh a document are identical for almost all Web Intelligence application developed by the EC. These ones are known by the DIGIT TESTCENTRE service.

The requester must describe each document to be refreshed with the following info:

| *Entitlement* | *Description* |
|---|---|
| Doc Id | This is the Id associated with the document. |
| Doc Name | This is the exact name of the document. |
| Doc Description | This is a brief description of the document |
| Corporate Document | This is the name of the Corporate Document link to access the list |
| Refresh parameters values | This is the operational values taken into account for refreshing the document. |

### *8.3.3. Test scenario*

| Entitlement | Description |
|---|---|
| Id | This is an id associated with the Test Scenario. |
| Name | This is a short name representative of the Test Scenario. |
| Description | This is a combination of document to be refreshed using the formatted notation (Refer to section 8.1) |

**What documents should be load tested**?

All the documents belonging to the corporate list must not be included in the test specification.

The requester must choose the most representative documents in point of view of their sizes and their frequency of refreshment.

The biggest documents have to be considered as well.

## 8.4. Performance test request form (Web services)

The performance test request form for Web services is almost identical to the performance test request form for J2ee Web application except that, instead of specifying actions to be executed, the requester must indicate Web services to be called.

Our load test tool uses the WSDL file to list all the services, methods, operations available and construct Soap requests based on the services selected.

The "URL of the Web services to be tested" and the "URL to access the WSDL" must be specified in the Application Information section.

Application Information, Contact Information and Technical Information parts are described in the "Performance test request form for J2ee Web application" section of this document.

### *8.4.1. Business Transactions*

Please refer to the section 8.1 of this document for the definition of a Business transaction, Atomic Action, Expected Average Response time.

In practical, each Business transaction represents a call to a Web service.

For each Web service, the exact name(s) of input parameter(s) and their values must be specified and referred to ASCII flat files. An accurate example of the soap request should be specified in the equivalent xml file.

If any parameter is constructed with special rules, it must be specified.

The notion of Think times is no more used with Web services but, instead, the requester must specify the number of calls to a web service per second, minutes or hour. This is done in the Test scenario section.

### *8.4.2. Test Scenario*

The requester may specify the concurrency of the Web services either by defining the number of concurrent users sending Soap requests or by limiting the number of calls per second.

**Here is a complete example:**

| Id | Atomic Action | Description | | |
|---|---|---|---|---|
| BT01 | WS01 (param1, param2) | call to Web Service 1 with input parameters contained in ws1.txt<br>or/and soap example in ws1.xml | 200 ms | 450 ms |
| BT02 | WS02 (param1) | call to Web Service 2 with input parameter contained in ws2.txt<br>or/and soap example in ws2.xml | 100 ms | 500 ms |
| BT03 | WS03 (param1, param2, param3) | call to Web Service 1 with input parameters contained in ws3.txt<br>or/and soap example in ws3.xml | 100 ms | 500 ms |

| | | |
|---|---|---|
| TC01 | Call to WS01 | BT01 |

| | | |
|---|---|---|
| TC02 | Call to WS02 and then to WS03 | BT02, BT03 |

| | | |
|---|---|---|
| TS01 | 10 users calling TC01 concurrently with 10 users calling TC02 | 10°*TC01 + 10*TC02 |
| TS02 | 10 calls per second to WS01 in concurrently with 10 calls per second to WS02. | TC01 (10/seconds) + TC02 (10/seconds) |

# 9. DESCRIPTION OF THE TEST CENTRE SERVICE

## 9.1. Organizational information

The Test Centre service is a small entity under the control of DIGIT A03 unit and located in DIGIT offices in Luxemburg.

It is a transverse service and is independent both from the development side (mostly the General Directorates) and production side (The DIGIT Data centre). This positioning is consistent with the common practices and recommendations related to the domain.

It can be contacted through a dedicated functional mailbox (DIGIT TESTCENTRE) and also has a dedicated group (DIGIT-SERV-TEST) in the Service Center Central Helpdesk System.

## 9.2. The service offered by Test Centre

The DIGIT Test Centre focuses on the test activities taking place after the software has been developed, the code is considered optimized and is functionally correct: the pre-production test category.

These pre-production tests are critical for the following reasons:

To guarantee the stability

To assess the performance

To avoid the potential perturbations on other Information Systems

To evaluate the necessary resources required to host the application

To determine the most efficient software configuration

The DIGIT Test Centre currently delivers the following pre-production testing services:

- Assistance for the definition of the pre-production tests specifications

  If requested by the DG, the Test Centre can help for the specification of the tests. Its added-value in this context is of course not in the business part of the specifications but to help in understanding how to efficiently request the service in a commonly understood and agreed way.

- Assessment and validation of the pre-production tests specifications

The Tester in charge will first make sure that the returned specification is adequate and realistic. She/He will optionally contact the DG to propose modifications when applicable.

- Test scripting
- Performance (load/stress/endurance) testing
- Capacity planning
- Bottleneck detection
- Tuning
- Test reporting

## 10. PROCEDURE FOR REQUESTING THE TEST CENTRE SERVICE

A stress test procedure is launch in two different cases:

- The DGs send an application deployment request and the CAB decide that the IS must be tested.
- The DGs wish to test a new version of an application that has already passed the stress test procedure.

The following flowchart details the procedures for the first case but the procedure is identical for the second case:

1. The DG is informed of the CAB decision.

2. The change coordinator will open a call to DIGIT SERV-TEST to initiate the request. He must include

   - ✓ The Technical information related to the stress test environment.
   - ✓ The document describing the architecture of the application.
   - ✓ The stress test specification form empty

   The TC can help you with the definition of the pre-production test specification and is responsible for the validation of this document.

3. In the meantime, the DC prepares the stress test environment and informs the DGs when it is ready.

4. The DGs requester should ask the Data Centre to deploy and configure the application on the stress tests infrastructure, including a realistic database (Maybe copy of production) and any other necessary assets.

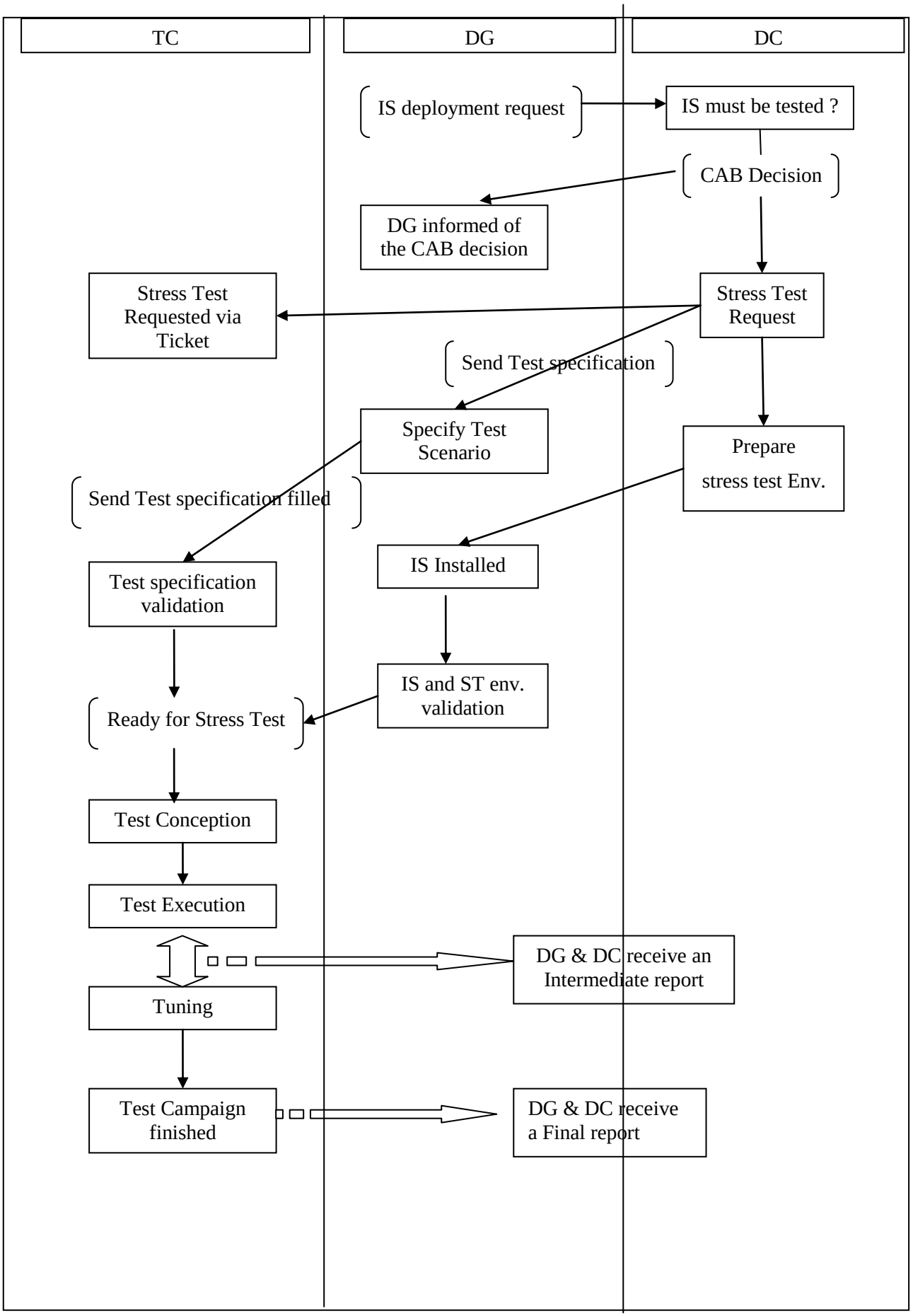
   Once the DGs have a confirmation that it is installed, he will have to validate this installation against the specified tests to make sure everything is fully functional.

   If this is ok, he informs the TC that the stress tests can be started.

The following highlights responsibilities associated to important tasks:

| Phase | Responsible | Contact in DIGIT | Functional mailbox |
|---|---|---|---|
| Prepare the ST environment | DIGIT ISHS service | assigned CCOR | DIGIT ISHS CCOR |
| Deploy application | DGs | assigned CCOR | |
| Configure application | DGs | assigned CCOR | |
| Verify application | DGs | | |
| Confirm application is OK | DGs | Assigned tester | DIGIT TESTCENTRE |

| | | | |
|---|---|---|---|
| Stress test application | DIGIT TESTCENTRE | Assigned tester | DIGIT TESTCENTRE |


TC
DG
DC
IS deployment request
IS must be tested ?
CAB Decision
DG informed of the CAB decision
Stress Test Requested via Ticket
Stress Test Request
Send Test specification
Specify Test Scenario
Prepare stress test Env.
Send Test specification filled
IS Installed
Test specification validation
IS and ST env. validation
Ready for Stress Test
Test Conception
Test Execution
DG & DC receive an Intermediate report
Tuning
Test Campaign finished
DG & DC receive a Final report

# 11. ROLES AND RESPONSIBILITIES

## 11.1. Role at the General Directorate

*Test Requestor (i.e. DG main contact)*

The Test Requestor is the DG initial single point of contact. She/he is responsible to fill and return the test specifications document to TC. The test specification form as well as the present document will be provided by TC.

*Functional contact*

The Functional contact is an individual knowledgeable on the business requirements and functional aspects of the application to be tested. She/he is supposed to known when and how to route requests to other DGs or external contacts having a more detailed view on parts of the application when applicable.

*Technical contact*

The technical contact is an individual knowledgeable both on the development and architectural aspects of the application to be tested. He is supposed to know when and how to route advanced technical request to other DGs or external contacts (typically architects and developers) having a more detailed view on components of the application when applicable.

## 11.2. Role at the DIGIT Data Centre

*Change coordinator*

The change coordinator is the DC initial single point of contact both for the DGs and the TC.

In the scope of the tests, she/he will guide the DGs for the installation of the Information System in the Load and Stress test specific infrastructure. She/He will also do the follow-up both for the Information System installation and the evolution of the tests. In addition, she/he is responsible for the internal dissemination of the test reports to the interested DC parties.

*System Administrators*

The System Administrators are the person in charge of the deployment and configuration of the base software infrastructure and application components.

The following is a non-exhaustive list of such components:

- Sun Solaris
- Windows
- Oracle database
- ColdFusion web application server
- Weblogic application server
- Business Objects server
- Other infrastructure components

They are also responsible for the assessment and validation of the configuration findings identified and set during the tests.

Their help will be solicited from time to time when the Test Centre has no other alternatives, tools or expertise to cover specific needs.

Examples of this are:

Modification of operating system parameters

Analysis and modification of a database either for engine parameters or assets optimisation

Analysis of ColdFusion database activity and code introspection

…

## 11.3. Role at the DIGIT Test Centre

*Service Manager*

The service Manager is responsible for the organisation of the Test Centre activities planning and resource allocation. He also is the contact person when a problem or question needs to be escalated above the Testers level.

*Testers*

The testers are initially responsible to provide assistance to the DGs during the test specification phase and to validate the test specification form.

In a second stage, they will implement the most efficient and adequate test strategy by implementing the scenario, execute them as specified, analyse and send intermediate and final reports on the findings and the test results both to the DG Test Requestor and the DC Change Manager.