[caTissue Suite v1.1.2 Release Details 1](#_Toc238973369)

[Introduction 1](#_Toc238973370)

[Release plan 2](#_Toc238973371)

[Technical details 2](#_Toc238973372)

[QA details 3](#_Toc238973373)

[Action items from previous release 3](#_Toc238973374)

[Test plan for v1.1.2 release 4](#_Toc238973375)

[caGrid Service Stress testing 10](#_Toc238973376)

[Performance testing 11](#_Toc238973377)

[Objectives of the Performance testing 11](#_Toc238973378)

[Summary of findings 11](#_Toc238973379)

[Test Infrastructure Setup and Environment 11](#_Toc238973380)

[Experiment details 11](#_Toc238973381)

[WU End user testing 13](#_Toc238973382)

[Appendix of Documents attached 13](#_Toc238973383)

# caTissue Suite v1.1.2 Release Details

## Introduction

The v1.1.2 Release is being developed by the TBPT Knowledge Center with the primary goal of resolving caGrid query performance and stability issues in caTissue. Issues that are fixed include:

1. Read timed out error while executing CQL or API query.

Bug#: <http://nagarajanlab.wustl.edu/bugs/show_bug.cgi?id=10018>

1. caTissue application server crashes with out of memory error while executing CQL queries.

<http://nagarajanlab.wustl.edu/bugs/show_bug.cgi?id=13573>

1. Cannot execute count or attribute based query through API or CQL. Such query throws java.lang.ClassNotFoundException.

Bug#: <http://nagarajanlab.wustl.edu/bugs/show_bug.cgi?id=9617>

The current status of this release is that the all testing is completed including functional, regression, stress, performance, and WU end user acceptance testing is completed.

## Release plan

**PSL/WU testing complete:** 08/21/09

**NCI testing schedule:** TBD

**Public release:** TBD

## Technical details

**Root causes:**

1. Hibernate lazy loading set to ‘false’ in many class-to-class associations led to poor database retrieve performance.
2. Unnecessary database retrieves in API query filtering business logic.

**What is Hibernate lazy loading?**

Consider a class ‘User’ with an associated class named ‘Address’ (i.e. User -> Address). Now:

1. If lazy loading is set to true, retrieving the User object will not automatically retrieve the Address object. Address will be retrieved only when a getAddress() call is explicitly performed on the User object.
2. If lazy loading is set to false, retrieving the User object via Hibernate will retrieve the Address object when the User object is retrieved.

**When to set lazy setting to true vs. false?**

There is no correct answer, and this is largely an application specific design consideration. Usually, one sets the lazy setting to false, if s/he is sure that the associated object will be needed along with the main object. In caTissue, the property is set to false in the following two cases:

1. If an associated object will always invariably be retrieved (e.g. Specimen to SpecimenCollectionGroup is set to false, because the SCG object is needed for most Specimen operations). Therefore, when one retrieves the Specimen object, s/he will get its associated SCG as well in one database retrieve.
2. If the associated object is a containment object of the main object (e.g. Address object is contained within a User object).

**Why does setting lazy to false cause problems?**

Given that someone executes a query to read all Specimen objects. This now has a cascading effect, where the service will read the associated SpecimenCollectionGroup, CollectionProtocolRegisration, etc. Therefore, time and memory for performing this query increases exponentially depending on the depth of the model. The impact of this is more profound if the query returns many objects.

**Why does this not cause a problem in the caTissue Advanced Query (executed through the API)?**

In advanced query, simple SQLs are used, not Hibernate.

**What is the solution for caTissue caGrid queries?**

We took advantage of the fact that CQL always returns only one object (called the target object in CQL) at a time. This means that even though caTissue was internally retrieving the data from the database for all the associated objects, it was returning just the data for the target object to the caGrid. Therefore, we modified the CQL to HQL processor to query explicitly for just the attributes of the target class specified in the CQL. This means that none of the associated data is retrieved from the database.

This applies only to the CQL queries and not for caCORE API-based queries. In case of caCORE API-based queries, one wants caTissue to return the associated objects also so that users can traverse through the associated classes.

**What is the impact of the code changes?**

The code changes are very local to the caGrid query and caCORE query API functionality (i.e. none of the User Interface business logic related code is impacted). The diagram below explains the Architecture block diagram as well as the modules changed.

**Note:** The changed modules are indicated by a red box.

## 

Figure 1: caTissue J2EE Architecture

## QA details

### Action items from previous release

1. Adding all caCORE API test cases into TMT (status: done – added 244 test cases)
2. Adding additional test cases for caGrid (status: done – added 25 caGrid test cases including authentication, PHI, privileges, and data based)

### Test plan for v1.1.2 release

Based on the code changes for the release, the following focus was decided for QA:

1. API and caGrid testing regression testing on all scenarios
2. Smoke testing of web application on all scenarios
3. caTIES regression testing
4. Stress testing of caTissue Suite v1.1.2 caGrid data service
5. Performance testing to compare caTissue Suite v1.1.2 caGrid data service vs. caTissue Suite v1.1.1 caGrid data service
6. WU end user acceptance testing of the caTissue Suite v1.1.2 caGrid data service

**Test matrix**

The following table summarizes the testing plan for v1.1.2 release. The test execution results are attached. Please refer to the Appendix for file names.

### 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module** | **OS** | **DB** | **Scenario** | **Type** | **Count** |
| caGrid | Linux | MySQL | Fresh | Functional | 27 |
| caGrid | Windows | MySQL | Fresh | Functional | 27 |
| caGrid | Linux | Oracle | Fresh | Functional | 27 |
| caGrid | Windows | Oracle | Fresh | Functional | 27 |
| caGrid | Linux | MySQL | Upgrade from v1.0 | Functional | 27 |
| caGrid | Windows | MySQL | Upgrade from v1.0 | Functional | 27 |
| caGrid | Linux | Oracle | Upgrade from v1.0 | Functional | 27 |
| caGrid | Windows | Oracle | Upgrade from v1.0 | Functional | 27 |
| caGrid | Linux | MySQL | Upgrade from v1.1 | Functional | 27 |
| caGrid | Windows | MySQL | Upgrade from v1.1 | Functional | 27 |
| caGrid | Linux | Oracle | Upgrade from v1.1 | Functional | 27 |
| caGrid | Windows | Oracle | Upgrade from v1.1 | Functional | 27 |
| caGrid | Linux | MySQL | Upgrade from v1.1.1 | Functional | 27 |
| caGrid | Windows | MySQL | Upgrade from v1.1.1 | Functional | 27 |
| caGrid | Linux | Oracle | Upgrade from v1.1.1 | Functional | 27 |
| caGrid | Windows | Oracle | Upgrade from v1.1.1 | Functional | 27 |
| API | Linux | MySQL | Fresh | Functional | 260 |
| API | Windows | MySQL | Fresh | Functional | 260 |
| API | Linux | Oracle | Fresh | Functional | 260 |
| API | Windows | Oracle | Fresh | Functional | 260 |
| API | Linux | MySQL | Upgrade from v1.0 | Functional | 260 |
| API | Windows | MySQL | Upgrade from v1.0 | Functional | 260 |
| API | Linux | Oracle | Upgrade from v1.0 | Functional | 260 |
| API | Windows | Oracle | Upgrade from v1.0 | Functional | 260 |
| API | Linux | MySQL | Upgrade from v1.1 | Functional | 260 |
| API | Windows | MySQL | Upgrade from v1.1 | Functional | 260 |
| API | Linux | Oracle | Upgrade from v1.1 | Functional | 260 |
| API | Windows | Oracle | Upgrade from v1.1 | Functional | 260 |
| API | Linux | MySQL | Upgrade from v1.1.1 | Functional | 260 |
| API | Windows | MySQL | Upgrade from v1.1.1 | Functional | 260 |
| API | Linux | Oracle | Upgrade from v1.1.1 | Functional | 260 |
| API | Windows | Oracle | Upgrade from v1.1.1 | Functional | 260 |
| caTIES | Linux | MySQL | Fresh | Smoke | 11 |
| caTIES | Windows | MySQL | Fresh | Smoke | 11 |
| caTIES | Linux | Oracle | Fresh | Smoke | 11 |
| caTIES | Windows | Oracle | Fresh | Smoke | 11 |
| caTIES | Linux | MySQL | Upgrade from v1.0 | Smoke | 11 |
| caTIES | Windows | MySQL | Upgrade from v1.0 | Smoke | 11 |
| caTIES | Linux | Oracle | Upgrade from v1.0 | Smoke | 11 |
| caTIES | Windows | Oracle | Upgrade from v1.0 | Smoke | 11 |
| caTIES | Linux | MySQL | Upgrade from v1.1 | Smoke | 11 |
| caTIES | Windows | MySQL | Upgrade from v1.1 | Smoke | 11 |
| caTIES | Linux | Oracle | Upgrade from v1.1 | Smoke | 11 |
| caTIES | Windows | Oracle | Upgrade from v1.1 | Smoke | 11 |
| caTIES | Linux | MySQL | Upgrade from v1.1.1 | Smoke | 11 |
| caTIES | Windows | MySQL | Upgrade from v1.1.1 | Smoke | 11 |
| caTIES | Linux | Oracle | Upgrade from v1.1.1 | Smoke | 11 |
| caTIES | Windows | Oracle | Upgrade from v1.1.1 | Smoke | 11 |
| PrivatePublic Migrator | Linux | MySQL | Fresh | Functional | 5 |
| PrivatePublic Migrator | Windows | MySQL | Fresh | Functional | 5 |
| PrivatePublic Migrator | Linux | Oracle | Fresh | Functional | 5 |
| PrivatePublic Migrator | Windows | Oracle | Fresh | Functional | 5 |
| PrivatePublic Migrator | Linux | MySQL | Upgrade from v1.0 | Functional | 5 |
| PrivatePublic Migrator | Windows | MySQL | Upgrade from v1.0 | Functional | 5 |
| PrivatePublic Migrator | Linux | Oracle | Upgrade from v1.0 | Functional | 5 |
| PrivatePublic Migrator | Windows | Oracle | Upgrade from v1.0 | Functional | 5 |
| PrivatePublic Migrator | Linux | MySQL | Upgrade from v1.1 | Functional | 5 |
| PrivatePublic Migrator | Windows | MySQL | Upgrade from v1.1 | Functional | 5 |
| PrivatePublic Migrator | Linux | Oracle | Upgrade from v1.1 | Functional | 5 |
| PrivatePublic Migrator | Windows | Oracle | Upgrade from v1.1 | Functional | 5 |
| PrivatePublic Migrator | Linux | MySQL | Upgrade from v1.1.1 | Functional | 5 |
| PrivatePublic Migrator | Windows | MySQL | Upgrade from v1.1.1 | Functional | 5 |
| PrivatePublic Migrator | Linux | Oracle | Upgrade from v1.1.1 | Functional | 5 |
| PrivatePublic Migrator | Windows | Oracle | Upgrade from v1.1.1 | Functional | 5 |
| Web app | Linux | MySQL | Fresh | Smoke | 48 |
| Web app | Windows | MySQL | Fresh | Smoke | 48 |
| Web app | Linux | Oracle | Fresh | Smoke | 48 |
| Web app | Windows | Oracle | Fresh | Smoke | 48 |
| Web app | Linux | MySQL | Upgrade from v1.0 | Smoke | 48 |
| Web app | Windows | MySQL | Upgrade from v1.0 | Smoke | 48 |
| Web app | Linux | Oracle | Upgrade from v1.0 | Smoke | 48 |
| Web app | Windows | Oracle | Upgrade from v1.0 | Smoke | 48 |
| Web app | Linux | MySQL | Upgrade from v1.1 | Smoke | 48 |
| Web app | Windows | MySQL | Upgrade from v1.1 | Smoke | 48 |
| Web app | Linux | Oracle | Upgrade from v1.1 | Smoke | 48 |
| Web app | Windows | Oracle | Upgrade from v1.1 | Smoke | 48 |
| Web app | Linux | MySQL | Upgrade from v1.1.1 | Smoke | 48 |
| Web app | Windows | MySQL | Upgrade from v1.1.1 | Smoke | 48 |
| Web app | Linux | Oracle | Upgrade from v1.1.1 | Smoke | 48 |
| Web app | Windows | Oracle | Upgrade from v1.1.1 | Smoke | 48 |
| PHI | Linux | MySQL | Fresh | Functional | 44 |
| PHI | Windows | MySQL | Fresh | Functional | 44 |
| PHI | Linux | Oracle | Fresh | Functional | 44 |
| PHI | Windows | Oracle | Fresh | Functional | 44 |
| PHI | Linux | MySQL | Upgrade from v1.0 | Functional | 44 |
| PHI | Windows | MySQL | Upgrade from v1.0 | Functional | 44 |
| PHI | Linux | Oracle | Upgrade from v1.0 | Functional | 44 |
| PHI | Windows | Oracle | Upgrade from v1.0 | Functional | 44 |
| PHI | Linux | MySQL | Upgrade from v1.1 | Functional | 44 |
| PHI | Windows | MySQL | Upgrade from v1.1 | Functional | 44 |
| PHI | Linux | Oracle | Upgrade from v1.1 | Functional | 44 |
| PHI | Windows | Oracle | Upgrade from v1.1 | Functional | 44 |
| PHI | Linux | MySQL | Upgrade from v1.1.1 | Functional | 44 |
| PHI | Windows | MySQL | Upgrade from v1.1.1 | Functional | 44 |
| PHI | Linux | Oracle | Upgrade from v1.1.1 | Functional | 44 |
| PHI | Windows | Oracle | Upgrade from v1.1.1 | Functional | 44 |
| caB2B End User | Linux | Oracle | Upgrade from v1.1.1 | Functional | 10 |

### caGrid Service Stress testing

**Server hardware:** Red Hat Enterprise Linux (kernal 2.6.9), RAM 2 GB, Pentium4 - 2.80GHz

**Database type:** Oraclewithdeidentified database of WU production instance

**Data size:** ~35K patients, ~300K specimens

**Types of testing:** Single user query, Concurrent user queries

**Other details:** Client and server were present within Persistent network

**Summary**

The goal of this exercise to test

1. Whether the caTissue web application or caGrid service will crash under stress
2. Whether the results are returned successfully

**Note:** The goal is NOT to find performance benchmarking numbers that can be published.

Therefore, all the test cases were using “worst case” queries (e.g., show all tissue specimens; show all molecular specimens, etc). We will be running more *real* end user queries on the WU QA setup soon and will report those performance numbers.

**Observations**

1. No query resulted in caTissue crashing. (which used to happen in v1.1 RC4)
2. Results were returned successfully (which used to give read time out in v1.1 RC4).
3. No memory leaks encountered when monitored using profiler and other monitoring tools.

**Query execution time details (all time in seconds)**

**Note:** Timings in a production environment are expected to be better than the ones reported below since the hardware configuration of production systems will be superior to the test servers at Persistent.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test scenario - Running the same query concurrently from different machines (Time in seconds is reported)** | | | | |
|
|
| **Query Name-->>** | **All Tissue specimens** | **All Fluid specimens** | **All Cell specimens** | **All Molecular specimens** |
| **No. Of Records Found** | **10000** | **10000** | **10000** | **9999** |
| **Machine 1** | 305 | 417 | 269 | 262 |
| **Machine 2** | 312 | 243 | 273 | 272 |
| **Machine 3** | 586 | 415 | 500 | 492 |
| **Machine 4** | 584 | 244 | 504 | 511 |
| **Average Time** | **446** | **329** | **386** | **384** |
|  |  |  |  |  |
|  |  |  |  |  |
| **Test scenario - Running different queries concurrently** | | | | |
|
| **Test Machine Name** | **Query Name** | | **No Of Records found** | **Time (seconds)** |
| **Machine 1** | All Molecular Specimens | | 9999 | 324 |
| **Machine 2** | All Cell Specimens | | 10000 | 598 |
| **Machine 3** | All Fluid Specimens | | 10000 | 597 |
| **Machine 4** | All Tissue Specimens | | 10000 | 264 |

## Performance testing

### Objectives of the Performance testing

The objective of this exercise is to benchmark the performance numbers of caTissue Suite v1.1.2 caGrid service against the caTissue Suite v1.1.1 caGrid data service. The tests are conducted on a hardware and database configuration which is very similar to WU caTissue production configuration.

### Summary of findings

The overall summary of the findings of this performance testing is that the newer version of caTissue is much faster than the older version. The performance improvement in terms of time ranges from 15% to 70% for queries which return data.

One of the major observations is that, in releases prior to v1.1.1, caTissue Suite caGrid Service as well as the caTissue Suite web application server would crash with Out of Memory error when any query returned more than 1000 records. This issue is now fixed in v1.1.2 build where queries as large as 20,000 records return data successfully without leading to server crash or any memory leaks.

### Test Infrastructure Setup and Environment

|  |  |
| --- | --- |
| **Processor** | 8 Processors, Intel Xeon CPUE5310 @  1.60GH, cache 4 MB |
| **Memory** | 8 GB |
| **Hard disk** | 128 GB (Partition used on which caTissue and caGrid are deployed) |
| **OS** | Distributer: CentOS release 5.2 , Kernel: 2.6.18 (for both Server and Database Machine) |
| **Database** | Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bi |
| **Data size** | Participants: 41,775  Specimens:463,139 |
| **Notes** | The server hardware, database and database contents are very similar to WU production environment. |

# Experiment details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **v1.1.2  (in secs)** | **v1.1 (in secs)** | **Improvement** | **No. of Records** |
| Return Molecular Specimen for CP short Title “Genetics of BWS” | 4 | 10 | 60% | 3 |
| Return Fluid Specimen that have been Thawed | 13 | 16 | 19% | 4 |
| Return Cell specimens that were fixed in formalin 30 minutes or less and were embedded in low melting point paraffin | 12 | 23 | 48% | 11 |
| Return DNA extracts derived from specimens where the cell fixative was not formalin | 4 | 10 | 60% | 1 |
| Return participants whose is having medical record from site ='Wellness Fair' | 4 | 10 | 60% | 1 |
| Return Molecular specimen where Parent specimen is of type 'Tissue' where CP title like '%Genetics%' | 4 | 6 | 33% | 2 |
| Return malignant prostate DNA specimens from African American participants who are on a CP with the word "Genetics" in the title | 15 | 18 | 17% | 3 |
| Return breast frozen malignant tissue specimens where tissue review event shows greater than 90% neoplastic cellularity | 16 | 73 | 78% | 48 |
| Return Tissue specimens which have been collected by the 'Lavage' procedure && that have been collected in an unacceptable quality | 12 | 14 | 14% | 1 |
| Return Patient for Tumor identified by needle Biopsy. | 13 | 44 | 70% | 78 |
| Return All Tissue Specimen | 114 | Server crash after 1020 seconds |  | 10000 |
| Return All Fluid Specimen | 99 | Server crash after 1020 seconds |  | 10000 |
| Return all Cell Specimen | 109 | Server crash after 1020 seconds |  | 10000 |
| Return All Molecular Specimen | 113 | Server crash after 1020 seconds |  | 9999 |

### WU End user testing

The WU end user testing was performed by Dr. Rakesh Nagarajan, Dr. Mukesh Sharma (caB2B project manager) and Madhumita Shrikhande (caB2B domain expert) by running end users queries using the following methodology:

1. Configuring the caB2B server to query caTissue Suite v1.1.2 caGrid data service
2. Executing a set of end user queries (listed below) on the caTissue Suite v1.1.2 caGrid data service using caB2B
3. Cross validating the results by querying the same set of queries using the caTissue Suite v1.1.2 Advanced Query interface

Following are the details of the testing done so far using the thick client/web application:

         Find all patients from whom Malignant Prostate tissue specimens are collected along with non-malignant specimens. (592 records returned)

         Find all patients from whom DNA and RNA specimens are collected. (3085 records returned)

         Find all patients from whom Urine and Whole Blood specimens are collected. (179 records returned)

         Find all patients from whom Malignant and non-malignant cell specimens are collected. (369 records returned)

         Find all patients from whom Malignant Bone marrow cell specimens are collected along with non-malignant specimens. (13 records returned)

# Appendix of Documents attached

1. **Result for KCPatch\_1.1.2\_RC2.xlsx** – This sheet contains the list of all the test results.
2. **Test Plans**

The table here specifies the file in the zip which contains the test cases for each module executed for this patch.

|  |  |  |
| --- | --- | --- |
| **Module** | **Type** | **Location of Test Plan** |
| caGrid | Functional | TestPlans\API\_caGrid\_caB2B.html |
| API | Functional | TestPlans\API\_caGrid\_caB2B.html |
| caTIES | Smoke | TestPlans\caTIES.html |
| PrivatePublic Migrator | Functional | TestPlans\Public\_Private\_Migrator.html |
| Web app | Smoke | All HTML files in TestPlans\WebApp folder:   * Biospecimen.html * MSR\_Admin.html * Ordering.html * Query.html * ShippingandTracking.html |
| PHI | Functional | TestPlans\PHI.html |
| caB2B End User | Functional | TestPlans\API\_caGrid\_caB2B.html |