[caTissue Suite v1.1.2 Release Details 1](#_Toc237236044)

[Introduction 1](#_Toc237236045)

[Release plan 1](#_Toc237236046)

[Technical details 1](#_Toc237236047)

[QA details 3](#_Toc237236048)

[Action items from previous release 3](#_Toc237236049)

[Test plan for v1.1.2 release 3](#_Toc237236050)

[caGrid Service Stress testing 3](#_Toc237236051)

# 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 development team has completed fixing the issues, and the release candidate is currently undergoing QA by the Persistent QA team and WU KC team.

## Release plan

**PSL/WU testing complete:** 08/14/09

**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.

The other feature we added is to configure the maximum number of records to be returned by caCORE API. This applies to all caCORE and caGrid queries. By default the caCORE API will return maximum 10000 records. This can be configured in CaTissueCore\_Properties.xml as a value of maxRecordsPercaCOREQueryAllowed property.

**What is the known issues?**

The known issue is OutOfMemory still may occur if the value maxRecordsPercaCOREQueryAllowed property is higher like 50000.

**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.

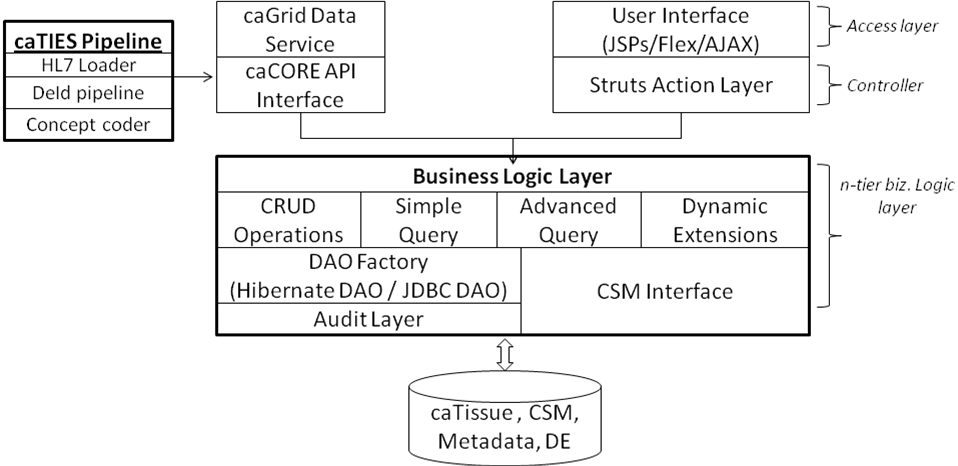


Figure 1: caTissue J2EE Architecture

## 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

**Test matrix**

The following table summarizes the testing plan for v1.1.2 release. As usual, all test results are recorded in TMT, and will be published along with the release.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module** | **OS** | **DB** | **Scenario** | **Type** | **Count** |
| Web app | Linux | MySQL | Clean | Smoke | 60 |
| Web app | Windows | MySQL | Clean | Smoke | 60 |
| Web app | Linux | Oracle | Clean | Smoke | 60 |
| Web app | Windows | Oracle | Clean | Smoke | 60 |
| Web app | Linux | MySQL | Upgrade from v1.0 | Smoke | 60 |
| Web app | Windows | MySQL | Upgrade from v1.0 | Smoke | 60 |
| Web app | Linux | Oracle | Upgrade from v1.0 | Smoke | 60 |
| Web app | Windows | Oracle | Upgrade from v1.0 | Smoke | 60 |
| Web app | Linux | MySQL | Upgrade from v1.1 | Smoke | 60 |
| Web app | Windows | MySQL | Upgrade from v1.1 | Smoke | 60 |
| Web app | Linux | Oracle | Upgrade from v1.1 | Smoke | 60 |
| Web app | Windows | Oracle | Upgrade from v1.1 | Smoke | 60 |
| Web app | Linux | MySQL | Upgrade from v1.1.1 | Smoke | 60 |
| Web app | Windows | MySQL | Upgrade from v1.1.1 | Smoke | 60 |
| Web app | Linux | Oracle | Upgrade from v1.1.1 | Smoke | 60 |
| Web app | Windows | Oracle | Upgrade from v1.1.1 | Smoke | 60 |
| caGrid | Linux | MySQL | Clean | Regression | 32 |
| caGrid | Windows | MySQL | Clean | Regression | 32 |
| caGrid | Linux | Oracle | Clean | Regression | 32 |
| caGrid | Windows | Oracle | Clean | Regression | 32 |
| caGrid | Linux | MySQL | Upgrade from v1.0 | Regression | 32 |
| caGrid | Windows | MySQL | Upgrade from v1.0 | Regression | 32 |
| caGrid | Linux | Oracle | Upgrade from v1.0 | Regression | 32 |
| caGrid | Windows | Oracle | Upgrade from v1.0 | Regression | 32 |
| caGrid | Linux | MySQL | Upgrade from v1.1 | Regression | 32 |
| caGrid | Windows | MySQL | Upgrade from v1.1 | Regression | 32 |
| caGrid | Linux | Oracle | Upgrade from v1.1 | Regression | 32 |
| caGrid | Windows | Oracle | Upgrade from v1.1 | Regression | 32 |
| caGrid | Linux | MySQL | Upgrade from v1.1.1 | Regression | 32 |
| caGrid | Windows | MySQL | Upgrade from v1.1.1 | Regression | 32 |
| caGrid | Linux | Oracle | Upgrade from v1.1.1 | Regression | 32 |
| caGrid | Windows | Oracle | Upgrade from v1.1.1 | Regression | 32 |
| API | Linux | MySQL | Clean | Regression | 240 |
| API | Windows | MySQL | Clean | Regression | 240 |
| API | Linux | Oracle | Clean | Regression | 240 |
| API | Windows | Oracle | Clean | Regression | 240 |
| API | Linux | MySQL | Upgrade from v1.0 | Regression | 240 |
| API | Windows | MySQL | Upgrade from v1.0 | Regression | 240 |
| API | Linux | Oracle | Upgrade from v1.0 | Regression | 240 |
| API | Windows | Oracle | Upgrade from v1.0 | Regression | 240 |
| API | Linux | MySQL | Upgrade from v1.1 | Regression | 240 |
| API | Windows | MySQL | Upgrade from v1.1 | Regression | 240 |
| API | Linux | Oracle | Upgrade from v1.1 | Regression | 240 |
| API | Windows | Oracle | Upgrade from v1.1 | Regression | 240 |
| API | Linux | MySQL | Upgrade from v1.1.1 | Regression | 240 |
| API | Windows | MySQL | Upgrade from v1.1.1 | Regression | 240 |
| API | Linux | Oracle | Upgrade from v1.1.1 | Regression | 240 |
| API | Windows | Oracle | Upgrade from v1.1.1 | Regression | 240 |
| caTIES | Linux | MySQL | Clean | Regression | 5 |
| caTIES | Windows | MySQL | Clean | Regression | 5 |
| caTIES | Linux | Oracle | Clean | Regression | 5 |
| caTIES | Windows | Oracle | Clean | Regression | 5 |
| caTIES | Linux | MySQL | Upgrade from v1.0 | Regression | 5 |
| caTIES | Windows | MySQL | Upgrade from v1.0 | Regression | 5 |
| caTIES | Linux | Oracle | Upgrade from v1.0 | Regression | 5 |
| caTIES | Windows | Oracle | Upgrade from v1.0 | Regression | 5 |
| caTIES | Linux | MySQL | Upgrade from v1.1 | Regression | 5 |
| caTIES | Windows | MySQL | Upgrade from v1.1 | Regression | 5 |
| caTIES | Linux | Oracle | Upgrade from v1.1 | Regression | 5 |
| caTIES | Windows | Oracle | Upgrade from v1.1 | Regression | 32 |
| caTIES | Linux | MySQL | Upgrade from v1.1.1 | Regression | 32 |
| caTIES | Windows | MySQL | Upgrade from v1.1.1 | Regression | 32 |
| caTIES | Linux | Oracle | Upgrade from v1.1.1 | Regression | 32 |
| caTIES | Windows | Oracle | Upgrade from v1.1.1 | Regression | 32 |

### 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 |