Xima Software
10653 South Riverfront Parkway,
Suite 200
South Jordan, UT 84095-3545, USA
sales@ximasoftware.com

# Xima Software selects OrientDB to provide flexible custom call detail reports

## 1. What is your organization and what does it do?

Xima was founded in 2007 and has become a leading telecommunications software company, providing simple and intuitive custom call reporting, call recording and call display solutions that make managing a team simple. Xima has grown to 48 employees, with 1,200+ resellers internationally, and since 2009 has averaged 102% annual growth. For the last three consecutive years, Xima has been a winner of the Best Place to Work award by Utah Business Magazine and listed as a Great Place to Work. In 2016 Xima was selected as a Top Workplace by the Salt Lake Tribune. Xima is a winner of the When Work Works award from the Society of Human Resource Management and Fortune Magazine named Xima the 22nd best place to work in America for Flexibility.

## 2. What are some of the key technical challenges that Xima faces?

Providing very flexible custom call detail reports required us to build a complex SQL-generating engine to query the data from our relational database. With larger data sets the complex queries began to become a significant performance problem for our larger customers. We could no longer rely on relational database performance tuning alone to satisfy these customer needs.

## 3. Tell us more about the application you developed and how graph DBs powered key parts of it. What was the goal of the application? What were important technical requirements?

We decided to build an in-memory reporting engine that would not be dependant on the actual database. We initialize the reporting engine by loading one hour sized blocks of call detail data spanning the timeframe required by the reports' criteria into an in-memory data grid (Hazelcast). Report results are then calculated using a map-reduce algorithm distributing the computation throughout the data grid.

Without the database dependencies, we began to explore alternative data stores that would be quick to load into our call detail data blocks. OrientDB was chosen to store Java objects (which we ultimately load into our data grid) that represent the call detail data. We then maintain a graph index of those call detail data objects. We have a graph of Year → Month → Day → Hour → Calls. The “Calls” contain the call detail data we then load into our in-memory data grid.

INDUSTRY
Telecommunications

CHALLENGE
Xima required a complex SQL-generating engine to query data from their relational database

SOLUTION
OrientDB provided a solution to store Java objects (representing call detail data) loaded into Xima's data grid

RESULT
With OrientDB in place, Xima is able to add new databases and classes (tables) more efficiently than with a traditional RDBMS

Xima Software
10653 South Riverfront Parkway,
Suite 200
South Jordan, UT 84095-3545, USA
sales@ximasoftware.com

4. Tell us how you found out about OrientDB?

OrientDB was discovered while looking online for Java-friendly document database storage options.

5. How did you ultimately settle on OrientDB as your vendor?

The ability to combine graphs with documents ultimately sold us on OrientDB.

6. How did you go about implementing OrientDB?

We obviously faced many challenges due to the inexperience of our developers with OrientDB. There were many challenges from the OrientDB implementation that we had to work around. For example, loading an object from the database in the Java API results in a proxy object. We heavily use Java serialization on our objects. Proxy objects don't fill our need, so we began “detaching” everything we pulled out of the database. We then found limitations and performance issues with the built-in detaching, so we had to implement our own deserialization.

We have been using the community edition, since we required the ability to redistribute the database to thousands of our customers. Since we're not paying for support, there were a few cases where we modified the OrientDB source code to fit our needs. For example, we run our other Windows Java services with Apache Procrun (this is the default service runner for Apache Tomcat). In order to enable OrientDB to run as a Windows service we made a couple of code changes to make it compatible with Procrun. We have also found a couple of bugs that we fixed along the way, submitting GitHub pull requests to benefit the entire OrientDB community.

We had several challenges dealing with exporting and importing data into OrientDB installations and instead began relying on only full backups and restores.

7. What were the results once your project was completed? Was the end goal achieved and initial challenges resolved? Were you satisfied with the results?

Our end goal has been achieved. We can load data quickly by time block. We do not have any specific metrics on time savings, revenue gains, or sales growth. Moving forward we anticipate quicker development. Now that OrientDB is in place we are able to add new databases and classes (tables) more quickly than we could with a traditional RDBMS.



OrientDB® by CallidusCloud