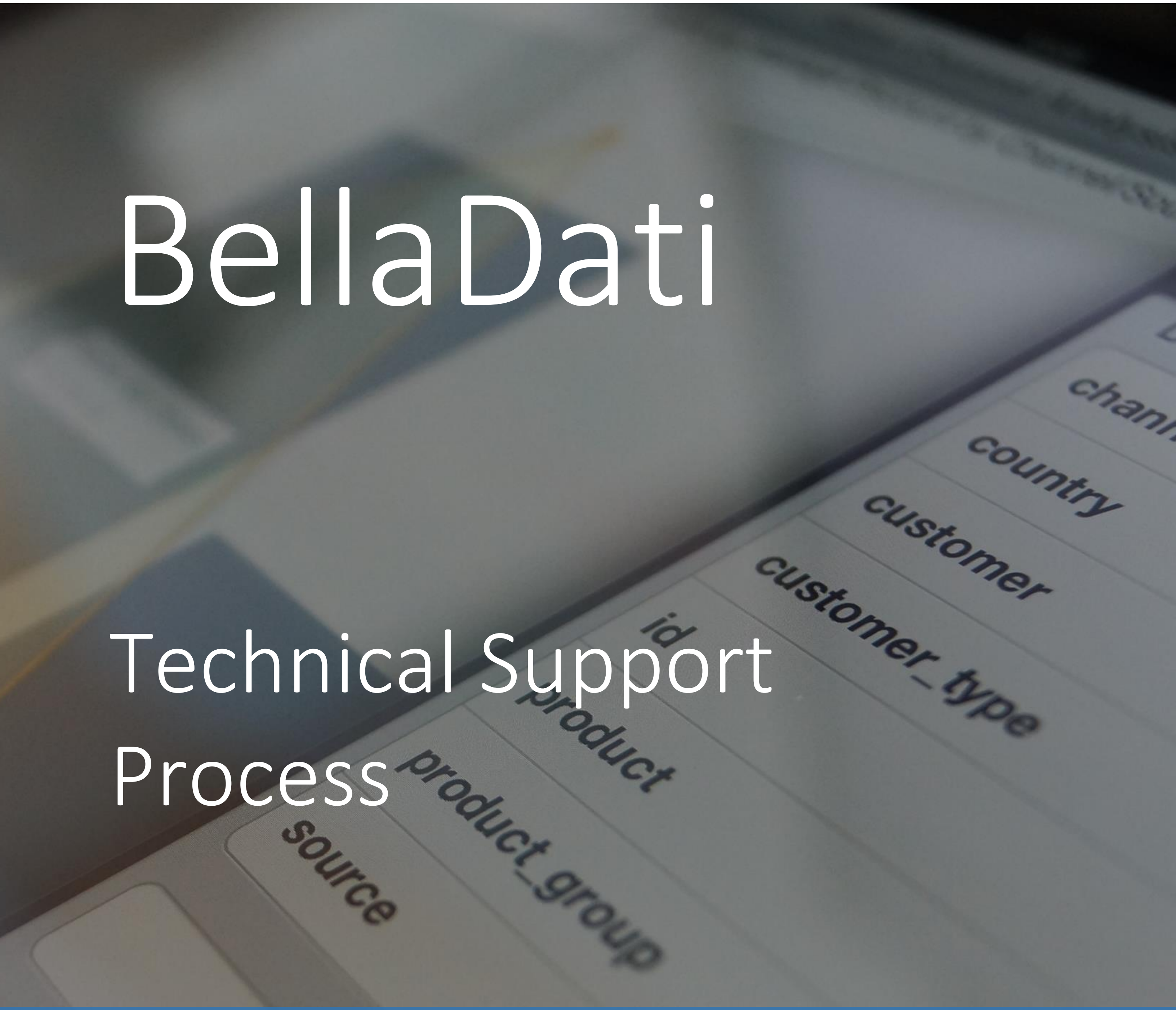

BellaDati

Technical Support Process

October 5th, 2017, rev. 4

Created by:
BellaDati HQ Asia
1003 Bukit Merah Central
Inno Centre, #07-06
Singapore, 159836

Document changes

Revision	Date	Author
1	15.2. 2015	Lubomir Micko
2	25.1. 2016	Miloslava Trginova
3	12.2. 2016	Lubomir Micko

Table of Contents

1 Purpose of the document

The purpose of this document is to define processes how BellaDati technical team communicates within the team and with Partner for Projects, eventually Customers, where the BellaDati support is requested. Usually it includes the following:

- Complex installations where BellaDati support is required
- Requirements for technical improvements and new features of BellaDati for delivery
- Fixing of the bugs discovered during the delivery process
- Embedding BellaDati into other products

Beside this, document defines BellaDati approach regarding delivery acceptance and process which follows the delivery.

2 Issues reporting

The main communication channel with BellaDati is BellaDati on-line Helpdesk (External JIRA). Each Partner is eligible for having own private workspace for reporting project-specific issues or requirements.

On-line Helpdesk system URL:

http://issues.belladati.com

Guidance of how to work with the Helpdesk is available here:

http://support.belladati.com/index/How+to+create+a+ticket

In case of emergency, Partners or Customer can escalate the tickets offline over emails or directly over the appropriate phone contacts defined in the communication matrix.

2.1 Issue definition

Each reported issue has a variety of associated information:

Name	Description
Issue type	Determines the type of the issue.
Summary	Provides brief and accurate summary of the issue
Description	Contains description of the issue, including steps how to reproduce the behavior in case of bug
Project	Defines the project the issue belongs to
Components	Components associated with the issue
Versions	Versions of the project which are affected by this issue
Environment	Environment in which issue occurs
Priority	Priority for being fixed or delivered
Status	Current status of the issue
Resolution	Resolution of the issue

2.2 Issue types

We recognize following issue types:

Name	Description
Bug	A problem which impairs or prevents the functions of the BellaDati product.
Improvement	An improvement or enhancement to an existing feature or task.
New feature	A new feature of the BellaDati product, which has yet to be developed.
Question	A question about BellaDati product functions or behavior.

2.3 Issue priorities

An issue has a priority level which indicates its importance.

Level	Description
Priority 1 - Blocker	Major problem affecting processing with severe impact on business
Priority 2 – Major	Problem which affects processing but does not have a severe impact on business
Priority 3 - Minor	Minor problem or general inquiry

2.4 Issue statuses

Each issue has a status, which indicates the stage of the issue. In the default workflow, issues start as being Open, progressing to In Progress, Resolved and then Closed. Other workflows may have other status transitions.

Status name	Description
Open	The issue is open and ready for the assignee to start work on it.
In progress	This issue is being actively worked on at the moment by the assignee.
Reopened	This issue was once resolved, but the resolution was deemed incorrect. From here issues are either marked assigned or resolved.
Resolved	A resolution has been taken, and it is awaiting verification by reporter. From here issues are either reopened, or are closed.
Tested	Considered correct according testing on testing environment.
Closed	The issue is considered finished, the resolution is correct. Issues which are closed can be reopened.

2.5 Issue resolutions

An issue can be resolved in many ways, only one of them being "Fixed". The defined resolutions are listed below. You can add more in the administration section.

Resolution	Description
Fixed	A fix for this issue is checked into the tree and tested.
Won't fix	The problem described is an issue which will never be fixed.
Duplicate	The problem is a duplicate of an existing issue.
Incomplete	The problem is not completely described.
Cannot reproduce	All attempts at reproducing this issue failed, or not enough information was available to reproduce the issue. Reading the code produces no clues as to why this behavior would occur. If more information appears later, please reopen the issue.

3 Testing strategy

BellaDati team follows the internal testing strategy which is defined in “BellaDati testing strategy” document. For better understanding the technical support process provided by BellaDati to Partners and Customers, the following chapter briefly describes the main points.

3.1 Testing goals

1. verify the implementation matches the specified requirement
2. verify the implementation is complete
3. verify the implementation behaves as expected, i.e. no bugs
4. ensure tests are always run in the same way
5. minimize effort and time required to run regression tests
6. leave room for testers to creatively explore application behavior in a structured way

3.2 Test types

BellaDati recognizes following types of testing.

Test type	Responsible	Note
Unit tests	Technical team	Automated in CI environment
Assembly tests	Technical team	Automated in CI environment
Functional tests	Testing team	
Integration tests	Testing team	
Regression tests	Business users, testing team	According to the testing scenarios
User acceptation tests	Business users	According to the business requirements

3.3 Test workflow

BellaDati test workflow consists of test planning, test specification, test design and test execution. Additionally, there can be exploratory testing.

3.3.1 Test planning

Time frame and deliverables for the test are determined, teams are set up or assigned. Exact schedule is subject to BellaDati development and working strategy and conditioned by the release management.

3.3.2 Test specification

Testers work with the requirement owner to find out about the intended application behavior.

To ensure testing goal #1, the tester must get their information from the requirement owner, not from a developer. A way to do this efficiently could be for the tester and developer to both meet with the requirement owner at the same time. Similarly, any clarifications from the requirement owner should be addressed to both testers and developers.

Based on the information from the requirement owner, the tester creates a written test specification to meet goal #4, covering all required application behavior to meet goals #2 and #3.

Depending on the type of test, the test specification may look differently. Tests involving the frontend often are lists of subsequent test steps, while API tests often contain just one API call with some test setup beforehand.

3.3.3 Test design and implementation

To achieve goal #5, regression tests are automated. This not only means testers can test a new build at the click of a button, but it also allows developers to run parts of the test suite when making changes and ensure their changes didn’t break any existing functionality.

For automated tests, this phase means implementing the test. For frontend testing, coordination with the developer may be required (e.g. to know the IDs of GUI elements) while API testing is entirely independent from development. Occasionally, existing tests need to be updated in this step if the requirement or implementation have changed.

BellaDati CI environment for automated tests is based on Jenkins and available at http://jenkins.belladati.com.

3.3.4 Test execution

Tests are executed against an application build. Automated test execution are done as part of a Jenkins build.

When doing manual testing, a tester needs to execute the steps described in the specification – testing scenarios. To achieve goal #4, the tester must execute tests exactly the way they were specified. If the specification is out of date, it needs to be updated to ensure test results can be reproduced in the future.

Not all tests need to be executed all the time, especially when testing manually. It is up to the testers to decide which tests they consider relevant.

3.3.5 Exploratory testing

In addition to regular tests, particularly for complex features it can be worthwhile to manually explore the application for unexpected behavior (goal #6). When an issue is found exploratively that wasn't detected by a regular, specified test, the tester adds it to the test specification and implements a test with the steps it takes to reproduce the problem.

3.4 Functional testing

For the BellaDati platform and applications built on BellaDati platform, 3 types of functional are relevant: API testing, frontend testing, and integration testing.

3.4.1 API Testing

The API offered by the server needs to behave the way it is specified. This is not only relevant for the frontend but also any other application using it, e.g. the mobile apps or 3rd-party clients.

Because there is no GUI interaction and the API can be defined very accurately, these tests are automated.

Tests have exactly two types of input:

- existing content in the database
- an incoming API requests

Similarly, they have two types of output:

- changes made to the database
- an API response

All four of these are controlled by automated tests in the following steps:

- Setup. The test inserts data directly into the database.
- Execution. The test makes an API request.
- Verification. The test verifies the new database contents and the API response.
- Teardown. The test clears the database.

3.4.2 Frontend Testing

BellaDati follows are two approaches to testing the frontend: individually and part of the whole application including a server. This section describes individual frontend tests. Tests involving both frontend and server are covered in Integration Testing.

Other than the API tests described previously, frontend tests involve interaction with the GUI over multiple test steps. Each test step can be an action to take or a verification to perform.

To test the BellaDati web frontend, Selenium framework is used. Using a simulated server, the tests control the data passed in to the frontend as well as verify any API requests being sent out. Because only a browser is required for test execution, these tests are parallelized and run in a distributed environment using Selenium Grid.

3.4.3 Integration Testing

Integration tests involve the whole application, including the server, the frontend, and other components. Integration tests may cover application behavior, but could also deal with the application installers, the mobile apps, or 3rd-party services.

The existing BellaDati Selenium tests fall into this category.

3.5 Non-functional Testing

Non-functional testing generally means tests that don't test for specific requirements but that are concerned with overall application behavior. Many of these cut across several requirements or even the whole application.

Goals for non-functional testing include:

- Cross-platform compatibility
- Usability
- Performance
- Security

3.5.1 Compatibility

Ensures that the system runs on all supported platforms. Can be automated, but there may be some challenges. Selenium can be used to run tests in different browsers; similarly, API tests can be run against a server deployed on different operating systems.

3.5.2 Usability

Requires manual testing, often exploratory. In-house testers can go through the application to ensure GUI components are in the right places, the right size etc. Additionally, 3rd-party testers (e.g. crowdsourced) can help verify that even users with little knowledge of the application can understand its workflows and use it effectively.

3.5.3 Performance

Both server and frontend performance can be relevant. For both modules, this can mean dealing with large amounts of data. On the server, it can also mean handling a large number of requests concurrently. These tests can be automated fully or run manually with the help of performance-testing tools.

3.5.4 Security

For BellaDati, the most relevant type of security testing is penetration testing, i.e. getting the application to do or reveal something it's not supposed to.

4 Technical support processes

4.1 Project/Delivery setup

This process applies when there is a new project or opportunity identified by the Partner or Customer.

1. Assign BellaDati team member roles for Project and get the same from Partner:
    a. Supervisor “S”
    b. Project manager “PM”
    c. Product functional consultant “PFC”
    d. Technical manager “TM”
    e. Tester “TESTER”
2. Kickoff meeting/call with the partner for technical team arranged by PM (sales part is already at reasonable stage). The Kickoff call should cover these topics:
    a. Mutual team member introduction and communication matrix setup.
    b. Introducing the Project scope and basic delivery characteristics.
    c. Gathering information from the Partner for completing of BellaDati Questionnaires by PFC. Only relevant information has to be completed, not complete questionnaires:
        i. BellaDati POC Questionnaire (see Appendix I)
        ii. Project high level overview (see Appendix II)
    d. File filled in questionnaires in project folders on Google Drive, notify team members. Clarification process is driven by technical team with the support of the PFC.
    e. Define the estimated time range of the project.
3. Create dedicated private space in the Helpdesk system and create appropriate user accounts for the Partner team member.

4.2 Requests for product improvements and new features

During the delivery process, Partner or Customer may raise requests for product improvements or new features implementation. This process follows these steps:

1. Main communication channel is BellaDati Helpdesk (see BellaDati Helpdesk). Partner specifies requirements containing the following main points:
    a. Feature expected behavior must be specified clearly, with prototype and with considering all related areas - roles and permissions, workflow, regression scenarios (if possible). The description of current behavior and expected behavior is preferred. Test case should be provided as well. We prefer to have sample BellaApp.
    b. Each feature must contain the severity and time for delivery confirmed by customer.
    c. requirements are collected and entered by the Partner in regular intervals (once in one or two weeks).
2. PFC will specify the requirement in BellaDati internal ticketing system (internal JIRA) and clarify with Partner and BellaDati development team.
3. PFC will inform the partner via External JIRA how many MH the resolution of the ticket will take and whether it is technically possible to implement. Putting issues on the high-priority list is possible and depends on the Partner agreement.

4. Partner confirms final request (prototype if required) and confirms the order/price for the ticket via Helpdesk system.
5. PFC confirms to the partner when ticket will be delivered via JIRA.
6. PFC informs Partner that ticket is completed via JIRA.
7. It is possible to share the RC1 or RC2 (release candidate) versions to the partners.
    a. nightly builds are not available for Partner. Nightly builds are purely development builds dedicated for continuous integration processes (automated testing, code coverage and other QA processes). Such builds are generated after every commit and are not stable and not dedicated for end-user testing. RC1 is built after the last internal issue is resolved, RC2 is tagged after the version passes the testing scenarios.
    b. If partner will be interested to contribute to the testing - especially when there will be features/issues related to partner's project delivery, RC1 or RC2 can be provided. Partners will be notified two days before official release will be ready.

4.3 Requests for embedding BellaDati into 3rd party products

Partners may request support for embedding BellaDati into other products. Usually there are following approaches:

- Re-brand BellaDati according your preferred look&feel
- Embed live visualizations with custom filters and basic analytical filters using iFrames, anonymous, authenticated or with SSO.
- Build own application utilizing BellaDati REST API and SDK
- Customizing BellaDati by using the Client API
- Create own visualizations or extend BellaDati Charts SDK

Basic procedures are described on BellaDati developers portal http://support.belladati.com/techdoc. If the Partner does not have knowledge or resources to Partner specifies the functional requirements

1. Partner describes the functional requirements and delivers lo-fi prototype of desired application.
2. BellaDati PFC evaluates the requirements and prototype and consults with technical team.
3. Depending on the complexity and integration level, BellaDati can offer the following solutions:
    a. Preparation of an example covering the specific case
    b. Remote guidance / assistance
    c. Implementing necessary functions in BellaDati (for fee)
    d. On-site implementation/integration (for fee)

4.4 Reporting and fixing bugs

Helpdesk system is used for bug fixing, essentially the same processes and procedures are used as for new functions requests.

1. Partner reports the bug to the Helpdesk system.
2. Partner ensures that a bug report (ticket) has relevant details or log file attached, otherwise it will be refused (if applicable, not possible for cloud).
3. Partner must evaluate and report the priority of bug and set up required date for fixing.

Level	Description
Priority 1 - Blocker	Major problem affecting processing with severe impact on business
Priority 2 – Major	Problem which affects processing but does not have a severe impact on business
Priority 3 - Minor	Minor problem or general inquiry

4. PFC will confirm whether it is a bug, will advise the workaround if possible and confirm expected date and version when bug fix will be delivered. Confirmed bugs (by BellaDati) are solved free of charge.

4.5 General rules

1. All tickets are communicated in English. Automated translators may be used however Partner must ensure that English translation is correct from technical point of view.
2. If Partner does not reply on tickets in timely manner, tickets will be closed due to inactivity. This will keep the helpdesk clean.
3. Tasks are internally in BellaDati assigned to development team, each member of development team is measured on timely and quality of task resolution.

Appendix I – Project overview template

Project overview

Customer	Customer name (http://www.website.com)
Partner	Partner name
PM Partner	Project Manager, email, phone
PM BellaDati	Project Manager, email, phone
Partner Team Members	Team member, email, phone
	Team member, email, phone
BellaDati Technical Team	Team member, email, phone
	Team member, email, phone
Project space (internal JIRA)	
Helpdesk project space (external JIRA)	http://issues.belladati.com/projectXYZ

Current Situation Description

Description of the current situation.

Expected/Proposed solution

Description of the expected/proposed solution.

Expected Delivery Timeline

Item	Timeline	Comment
Project opening stage	?	Provide information on suitability of BellaDati Discuss and confirm proposed solution
Expected project kick-off date (estimated)	?	
Expected project pilot date (if applicable, estimated)	?	
Expected project go-live date (estimated)	?	

Appendix II – BellaDati Questionnaire Example

Partner / Customer	
PoC Identification	
Date	
Notes	

Contacts

Name	**E-mail**	**Phone Number/Skype**	**Role**
(Please provide contacts for people responsible for technical part and analytical part of project. If it is embedded analytics project or integration include Product Manager and Developer contacts as well.)			
Product/Project Manager			
Data analyst			
Technical support engineer			
Developer engineer			
Sales manager			

Operation / Usage

Estimated amount of data to be analyzed *(e.g. data amount, number of records, growth rate, …)*	
Import frequency, import methods *(Every hour, daily, incremental import, etc.)*	
Est. number of concurrent imports	
Est. number of data sets	
Est. number of reports	
Est. number of users	
Est. number of concurrent users	
Testing/Staging environment	
HA environment	
Load balancing (front-end)	
Mobile application	
REST API	
Est. usage of REST API	

IoT Data Collector

Estimated number of devices sending the data	
Data sending frequency *(How often does the sensor send the data)*	
Est. size of the data message sent to BellaDati	

Data sets & Data sources

Data source type *(SQL Database, NOSQL database (MongoDB), XML/A Server (OLAP/Analytic services, SAP BW, MSSQL), Network resource (FTP, URL, SAMBA), HTTP service (REST, SOAP, …), Flat file (CSV, text), Structured file (XML, XLSX), ZIP Archive, Facebook, Google Drive, Google Analytics, SFDC, LinkedIn, Twitter, Amiando, Zendesk, Changegear, Other)*	
Data source vendor *(e.g. Oracle, SAP, Hadoop, Facebook)*	
Version *(application for database, XML/A, web services, e.g. Oracle 11g)*	
Authentication mode *without authentication* *Credentials passing* *Basic HTTP* *oAuth (1, 2, xAuth)* *SSL client authentication*	
Connection details (e.g. ports, users, constraints, VPN, firewalls, proxies)	
Transformation needed	
Data level permissions (vertical)	
Horizontal permissions *(attributes, indicators, data sets)*	
Dynamic permissions evaluation *(e.g. based on the lookup table)*	

Reports

Est. number of indicators (facts)	
Est. number of attributes	
Required visualizations *Chart (line, bar, stack bar, scatter, radial, heat map, thermometer, funnel, speedometer, candle, tree map, bullet, Gantt)* *Table* *Map (points, shapes)* *KPI Labels* *Custom content*	
Average visualizations per report	
Basic calculations *(e.g. sum, avg, min, max, rank, counts, …)*	
Advanced calculations, formulas *(e.g. regression, percentile, rates, distributions, etc)*	
Pre-defined filters *(user filters, view filters, custom filters)*	
Custom attributes aggregations *(e.g. grouping by specific attributes)*	
Custom date/time, numbers formats	
Styles customization *(e.g. colors, logos, tables styles, ..)*	
PDF export customization *(e.g. colors, logos, tables styles, ..)*	
Multilingual reports	
Parameterized reports (drill-trough)	
Linking values with external systems	
Embedding into external system	
Alarms	

Domain/user provisioning

ActiveDirectory/LDAP authentication	
ActiveDirectory/LDAP attributes mapping *(e.g. automatic filters, user groups)*	
Single-Sign-On (SSO) Centralized Auth. Server Federated Login Other	
3rd Party application authentication *(Facebook, SalesForce, LinkedIn, Google, …)*	
REST API Domain/User provisioning	
Other technology related requirements	

Proof of concept

Provide sample data (provide URL with data file or connection to DB, eventually specify webservice) **Provide data in raw format =** delimited txt file (CSV), raw data in Excel (not modified in Excel)	
If you provided DB access - Include SQL command structure to fetch desired data from DB	
Short description and or mock-up picture of desired reports (or attach samples of already existing reports)	
Specification of sample data Description of attributes (drill-down) and indicators Relations between tables in DB if needed, Explain formulas to calculate KPI's from raw data (that you used in original reports)	