Impact Analysis Tool Dev Documentation

**Content**

[1. Purpose 1](#_Toc480294155)

[2. Indented audience 1](#_Toc480294156)

[3. Prerequisites 1](#_Toc480294157)

[4. Architecture 2](#_Toc480294158)

[Program flow (Activity diagram) 2](#_Toc480294159)

[Client - Server Architecture 4](#_Toc480294160)

[Client - Server Communication 4](#_Toc480294161)

[5. The frontend 5](#_Toc480294162)

[Folder structure 5](#_Toc480294163)

[Dependencies 5](#_Toc480294164)

[The code structure 6](#_Toc480294165)

[6. Further development and production deployment 6](#_Toc480294166)

[Installation requirements 6](#_Toc480294167)

[How to develop locally 6](#_Toc480294168)

[How to deploy 7](#_Toc480294169)

[7. Help 7](#_Toc480294170)

# Purpose

This document describes how the impact analysis tool (short: IA tool) has been developed and how it can be changed or extended. It also shows the architecture of the IA tool.

# Indented audience

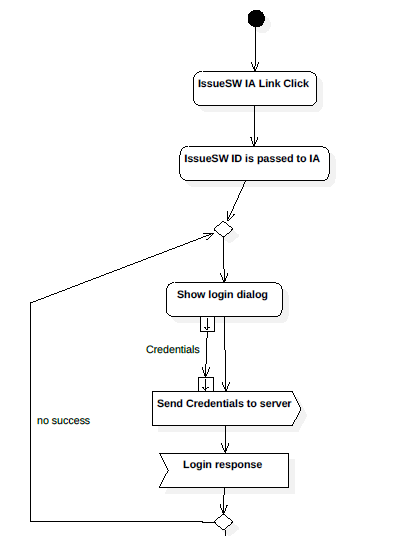
Knowledge of the tools architecture is particularly necessary for people who want to set up, extend or customize the impact analysis tool.

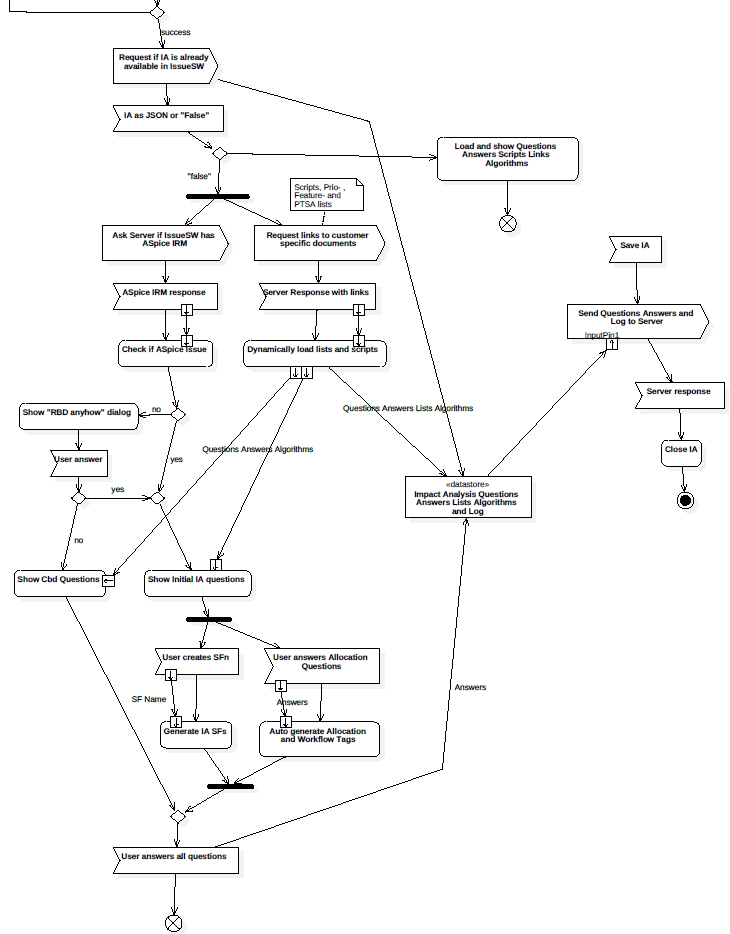
# Prerequisites

Before you read the how to or the documentation chapter please make sure that you are familiar with the tool, its requirements and features.

# **Architecture**

## Program flow (Activity diagram)





## Client - Server Architecture

The application consists of two parts: The server (backend) and the client (frontend).

* IA-Tool Server

Currently implemented in C# on the IP-Tool Server by Dominik Reiter. The IP-Tool offers two servers: One for development and one for production.

**Dev-URL:** <https://wi0www02.emea.bosch.com/BuggyLockedTest/IAToolService/>

**Production-URL:** <https://wi0www02.emea.bosch.com/BuggyLocked/IAToolService/>

* RQ1-Server

The IA-Tool Server communicates with the RQ1 Server via OSLC to retrieve and store data. The basic operations can be extracted from the chapter “Client-Server Communication”.

* Client

The client program is executed within the browser and is implemented in HTML, CSS and JavaScript. The exact dependencies and architecture is described in the chapter “The frontend”. The files are served and the tool is accessible from the following URL:

<https://wi0www02.emea.bosch.com/ImpactAnalysis>

While most of the IA tools functionality is directly implemented in JavaScript and executed on client side from the Browser, the basic operations for communication with RQ1 are implemented on server side. Communication between the client and the server is done 100% with AJAX at the moment to decouple the basic IA functionality from the storage of the data. The format for exchanging data between the server and client is JSON. In this way it is easily possible to completely replace and test the backend without changing the frontend (just change the variable ia.server\_url). This has been done because in future we do not want to store the Impact Analysis in RQ1 as attachment, but want to use a long-term achievable database.

## Client - Server Communication

***Request*** ImpactAnalysis\_Login

***Input*** User Credentials

***Server action*** Check if credentials are valid

***Output*** true/false (in future it would be good if a cookie is returned, so that there is no need to

store the credentials as local JavaScript variables)

***Request*** ImpactAnalysis\_IssueInAspiceProject

***Input*** IssueSW-ID as URL-Param

***Server action*** Server checks if there is an IRM connected to the Issue, which belongs to a project

which has the tag ***<ReqBasedDev>true</ ReqBasedDev>*** set

***Output*** true / false

***Request*** ImpactAnalysis\_IssuePoolProject

***Input*** IssueSW-ID as URL-Param

***Server action*** Server extracts the Pool Project ID of the Issue SW from RQ1

***Output*** Pool Project ID

***Request*** ImpactAnalysis\_GetImpactAnalysis

***Input*** IssueSW-ID as URL parameter

***Server action*** The server checks if the issue has an attachment with the name

„ImpactAnalysis\_[Issue SW ID]\_[number].json“. If yes it downloads and parses it.

***Output*** null – if the issue has no „ImpactAnalysis\_[Issue SW ID]\_[number].json“ attachment

The content of the file as JSON.

***Request*** ImpactAnalysis\_UpdateImpactAnalysis

***Input*** Issue ID as parameter and IA-Data as JSON

***Server action***

* Server creates “ImpactAnalysis[Issue SW ID].json” file and uploads it to the Issue SW. If file already exists the server attaches a “\_[Consecutive number]”.
* Server stores/overwrites the created file on the IA Tool server. <https://wi0www02.emea.bosch.com/ImpactAnalysis/files/>
* Server writes the allocation and allocation-comment tags to the Issue SW based on the JSON content the client sends

***Output*** success / error

**Additional Requirements:**

* For each new Issue SW the RQ1 server automatically attaches a link to the IA tool within the internal comment including also the Issue SW Id in the parameter.

# The frontend

## Folder structure

* css contains the application specific css files
* customer\_specific contains the department specific files, structured by the Pool project

ID. This includes department specific questions, code and csv files.

* default-files contains the default files if no customer specific files have been found
* docs contains all the IA tools documentation
* examples contains impact analysis examples
* img contains image files relevant for the IA tool frontend
* js contains the application specific JavaScript files
* libs contains the external JavaScript libraries
* tests contains JavaScript unit tests
* index.html contains the HTML and is the starting point for the application
* web.config contains the IIS Server configuration
* gulpfile.js used for automating tasks (like production deployment)

## Dependencies

Due to the Bosch Firewall the dependencies have not been managed via npm, but instead been directly downloaded and stored in the /libs folder.

* **jQuery**

Use for DOM manipulation and AJAX communication with the Server.

* **Bootstrap**

Used for providing a responsive and elegant frontend design.

* **PapaParse**

Used for parsing CSV files and turning them into JavaScript objects.

* **QUnit**

Used for some simple unit tests.

## The code structure

Currently refactoring…

# Further development and production deployment

## Installation requirements

For adding new features to the IA-Tool server please contact the ECB-Tool Team (Dominik Reiter), because this team handles the IA-Tool server implementation.

For further development of the IA-Tool features you need to install the following tools:

* Git
* NodeJS incl. the following configuration:

npm config set https-proxy http://user:password@rb-proxy-unix-de01.bosch.com:8080

**Set the proxy configuration as system/user environment variables (uppercase is important!):**

HTTP\_PROXY http://<user>:<pw>@rb-proxy-de.bosch.com:8080

HTTPS\_PROXY http://<user>:<pw>@rb-proxy-de.bosch.com:8080

* NPM modules

npm install

* A local npm http server. You can install one globally by typing the following into the command line:

npm install http-server -g

* The gulp taskrunner which is used for production deployment

npm install gulp-cli -g

* A good editor for writing Code (preferable with JSLint support)
* A browser (Firefox)

## How to develop locally

* Add the URL **wi0www02.emea.bosch.com/ImpactAnalysis** as network directory
* Update the constant **server\_location** in the file **gulpfile.js**
* Go to the root directory of the application (locally, not on the server)
* Start the local http server
* Visit one of the displayed urls (e.g.: <http://127.0.0.1:8080>). You should be able to see the Impact Analysis tool.

**Attention:** Although you can develop locally and the tool communicates with the test server, the test server itself gets its data from the productive RQ1 database. This means that you can use real RQ1 issues.

## How to deploy

1. **Extensively test the application (at the moment you have to do it manually)**

Deploying a version with a major bug annoys the users. Test at least all the features you have changed/added.

1. **Create a new version**
   1. **Code**

If you have changed the code update the member **ia.TOOL\_VERSION** in the file initialize.js.

* 1. **Questions**

In changed all text.js files (default or customer specific) you need to update the member **ia.QUESTION\_VERSION**.

* 1. **Git**
     + **Commit**

git add -A

git commit -am “commit for version X”

* + - **Create tag**

git tag -a v1.4 -m "version 1.4"

1. **Deploy to production**

You can deploy to the production server by simply executing the command:

**gulp deploy**

The deployment script will:

* + - Zip the current production and development folders and store them into the files/old directory on the server
    - Overwrite all existing files on the server with the files in the development folder

Attention: You will need to manually copy the customer specific files. See below!

1. **Update customer specific files**

The customer specific files are not copied, because someone might have changed them directly on the server. Therefore you will need to copy them manually.

# Help

In case you need further information, simply contact me: [hermann.wagner2@bosch.com](mailto:hermann.wagner2@bosch.com).