FD - Number Analyzer

Content

History

Version	Date	Adjustments
A	2006-10-16	First version. (MHATU)

1 Introduction

The purpose of this document is to explain the internal structure and functions of the Number Analyzer component. The Number Analyzer functions are specified in [1].

2 Function Structure

2.1 Overview

This picture shows which external modules the number analyzer depends on.

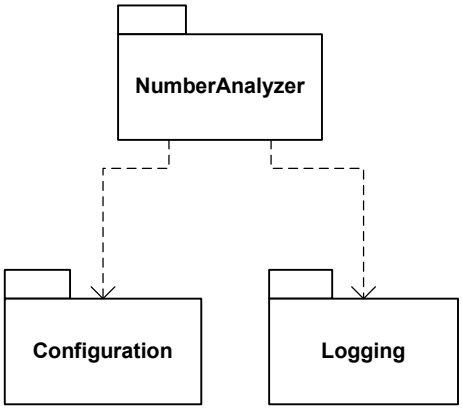

Figure 1 Component dependencies

The following external classes/interfaces are utilized by the number analyzer:

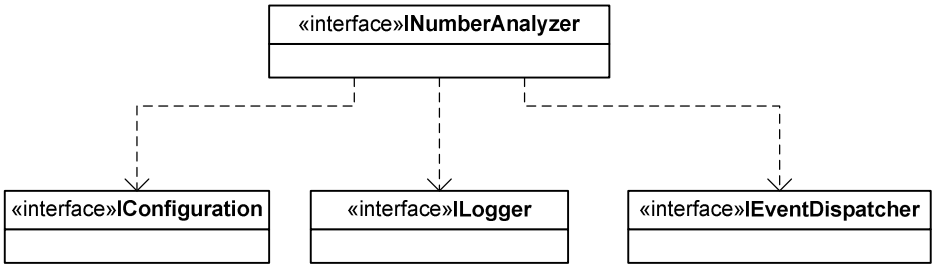

Figure 2 External interfaces dependencies

- Configuration is used to read the configuration needed for the number analysis function.
- Logging is used for error and debug messages.
- Event Dispatcher is used to receive configuration changed events.

2.1.1 External API

The Java Reqexp API [3] is used to perform the pattern matching of numbers. From Java 1.5 this API is included in the standard Java API.

2.2 Modules

This chapter explains the different modules and classes and what functions they do.

The component consist of one package: *com.mobeon.masp.numberanalyzer*

See also the class diagram in [2].

2.2.1 AnalysisInput

This class implements the IAnalysisInput interface, contains the following info: The number that is going to be analyzed, the name of the rule and information containing a region code. See [1] also.

2.2.2 NumberAnalyzer

Implementation of the INumberAnalyzer interface. This is the class that performs the number analysis. It analyzes a number with the rules defined in the Rule class, see below.

2.2.3 Rule

A Rule has a name, an input expression and a return expression. A Rule can also contain a List of Rule objects (sub rules). The rules are configured in a special configuration file.

Here is an example on a rule named INBOUNDCALL in the configuration file. It has two sub rules "NumberLength" and "All". Each of the subrules has an input expression and a return expression. The subrules are checked in the order they appear in the rule. I.e. the "NumberLength" is checked before the "All" rule.

```
<rule name="INBOUNDCALL">
    <subrule name="NumberLength">
        <input expr=""/>
        <return expr="4,12"/>
    </subrule>
    <subrule name="All">
        <input expr="^(1[09])([0-9]{2})$"/>
        <return expr="466010$i1$i2"/>
    </subrule>
</rule>
```

2.2.3.1 Input expression

This is a regular expression string which is matched against the number to analyze. If the expression matches, this rule applies for the number analysis.

2.2.3.2 Return expression

This is an expression that specifies what should be returned as result after the number analysis.

There are three different types of return expressions:

2.2.3.2.1 Group return expression

It can contain identifiers to the regular expression pattern from the input expression. The identifier has the following format $i<group id>. The group id relates to the number of the regular expression group. For example in the following pattern *^(555)(\d*)$* $i1 relates to the (555) part and $i2 relates to the (\d*) part. Also See [3]

The return expression may also contain strings that should be applied to the resulting number.

Ex on expected result from on different input patterns and corresponding return expressions:

Input: ^(1[09])([0-9]{2})$

Return: 4660$i1$i2

The number *1999* will be evaluated to *46601999* (4660 + 19 + 99)

2.2.3.2.2 Length return expression

It is used to check correct length on a number. The expression contains of a min value and a max value separated by comma. Ex:

Input:

Return: 4, 12

Note that the input expression is ignored when this return expression is used.

2.2.3.2.3 Block return expression

It is used to say that the number is blocked if the rule matches the number. Ex:
Input: ^(555)
Return: Block

The number 555161074 will be blocked because it starts with 555 and the return expression has the “Block” string.

2.2.3.3 Regioncodes

The AnalysisInput may contain information about the region of the number. Some special rules may be configured to match this region information. And if a match that region code is applied to the number. Ex:

Configured region codes: *060,061,062*

Region code information: *061161070*

Number to be analyzed: *161074*

Evaluated number: *061161074*

2.2.4 NumberAnalyzerConfiguration

This class contains functionality to read the configuration file used in the number analyzer component. It reads the file and puts the Rules into a map which can be used by the NumberAnalyzer class.

The configuration file is named numberanalyzer.xml and the schema definition file is similarly named numberanalyzer.xsd.

2.2.5 NumberAnalyzerException

This is an exception class that is thrown when some error occurred during the number analysis. For example the number did not match any rule or the number has incorrect length.

3 Function Behavior

This chapter describes the gory details of how the number analysis works.

3.1 Initialization

The Number Analyzer is as all the components in MAS initialized via the Spring Framework. The external interfaces are injected into the NumberAnalyzer class when it is created.

These methods exist for this reason:

public void setConfiguration(IConfiguration configuration)

public void setEventDispatcher(IEventDispatcher eventDispatcher)

The NumberAnalyzer class exists as a singleton object in MAS.

3.2 Number Analysis

This sequence diagram shows how the number analysis works.

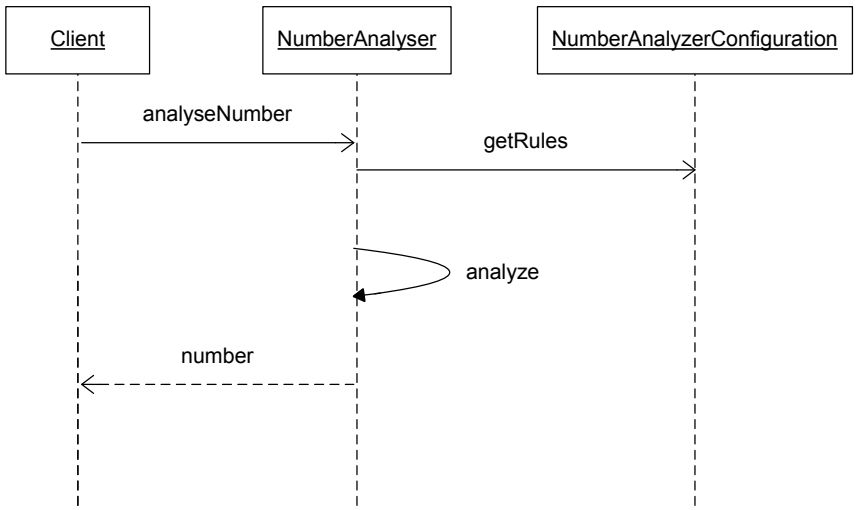

Figure 3 Number analysis.

1. The client calls the analyzeNumber method with AnalysisInput as argument.

2. The NumberAnalyzer class takes the name on the rule and fetches the corresponding Rule from the NumberAnalyzerConfiguration rulemap.
3. The number is then analyzed with that Rule.
4. If a match is found the return expressions are applied to the number string and the number is returned.

If the number does not match a NumberAnalyzerException will be thrown. See below:

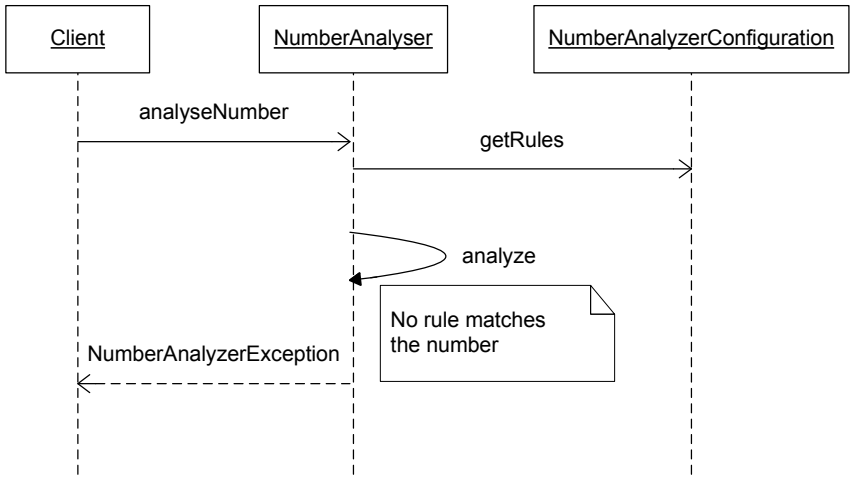

Figure 4 Number analysis that throws exception

1. The client calls the analyzeNumber method with AnalysisInput as argument.
2. The NumberAnalyzer class takes the name on the rule and fetches the corresponding Rule from the NumberAnalyzerConfiguration rulemap.
3. No rule matches the number.
4. A NumberAnalyzerException is thrown. It contains information about what went wrong.

3.3 Configuration reload

MAS do not have to be restarted in order to update the configuration for this component.

Number Analyzer is registered as an Event receiver in the EventDispatcher interface. When a ConfigurationChanged event is received the Number Analyzer updates the configuration in the same way it did at start-up. The stored rules are cleared and new ones are read from the new configuration.

3.4 Command line tool

The number analyzer component has also a test tool which can be used to validate the regular expressions and rule design of the configuration file.

It is a command line interface that runs with this command:

```
java -classpath $CLASSPATH
com.mobeon.masp.numberanalyzer.NumberAnalyzer –f <filename> -r <rule>
-n <number> -a <aninumber>
```

The options are explained here:

-f name and location of the numberanalyzer.xml to test

-r name on the rule to test

-n number to test

-a Ani number that includes region code information (optional)

This tool can be wrapped in a shell-script.

4 References

[1] FS – Number Analyzer
12/FS-MAS0001 Uen

[2] FD – Visio diagram
12/FD-MAS0001 Uen

[3] Java Regexp API
http://java.sun.com/j2se/1.5.0/docs/api/index.html

5 Terminology

API	Application Programming Interface

6 Appendix

Example on numberanalyzer.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="numberanalyzer.xsd">
    <numberanalyzer>
        <rule name="INBOUNDCALL">
            <subrule name="NumberLength">
                <input expr=""/>
                <return expr="4,12"/>
            </subrule>
            <subrule name="All">
                <input expr="^(1[09])([0-9]{2})$"/>
                <return expr="466010$i1$i2"/>
            </subrule>
        </rule>
        <rule name="CALLEROUTDIAL">
            <subrule name="NumberLength">
                <input expr=""/>
                <return expr="4,6"/>
            </subrule>
            <subrule name="Office">
                <input expr="^(1[09])([0-9]{2})$"/>
                <return expr="$i1$i2"/>
            </subrule>
            <subrule name="Local">
                <input expr="^([1-9])([0-9]{4,5})$"/>
                <return expr="$i1$i2"/>
            </subrule>
        </rule>
        <rule name="BLOCK">
            <subrule name="blocktest">
                <input expr="^(555)"/>
                <return expr="Block"/>
            </subrule>
        </rule>
        <rule name="test2">
            <subrule name="NumberLength">
                <input expr=""/>
                <return expr="6,6"/>
            </subrule>
            <subrule name="Local" regioncoderule="RegionCodes">
                <input expr="(\d*)$"/>
                <return expr="$i1"/>
            </subrule>
        </rule>
        <rule name="RegionCodes">
            <input expr="060,061,062"/>
        </rule>
    </numberanalyzer>
</configuration>
```