On the Development of a CSPA Error Correction Service: Design and Implementation Issues

Donato Summa\(^1\) (donato.summa@istat.it), Marco Silipo (silipo@istat.it), Monica Scannapieco (scannapi@istat.it), Diego Zardetto (zardetto@istat.it) and Mauro Bruno (mbruno@istat.it)

**Keywords:** CSPA, Statistical Services, SOA

1. **INTRODUCTION**

CSPA (Common Statistical Production Architecture) [1] is a template architecture for supporting the industrialization of Official Statistics production processes. CSPA includes some specifications intended to define interfaces of services in a standard way, with a focus on service inputs and outputs. In the paper, we describe an experience related to the implementation of the CSPA service “Error Correction” service, realized within the CSPA Implementation Project, an international project within the High Level Group Modernization program [2]. The realized service wraps functions that are offered by the R package “rspa” developed at Statistics Netherlands [3].

1.1. **The CSPA Concept**

CSPA provides template architecture for official statistics, describing:

- What the official statistical industry wants to achieve
- How the industry can achieve this, i.e. principles that guide how statistics are produced
- What the industry will have to do, compliance with the CSPA

The principal aims of this template architecture are: (i) Provide guidance for building reliable and high quality services to be shared and reused in a distributed environment (within and across statistical organizations); (ii) Enable international collaboration initiatives for building common infrastructures and services, and (iii) Foster alignment with existing industry standards such as the GSBPM and GSIM (Generic Statistical Information Model).

CSPA is based on Service Oriented Architecture (SOA). Statistical services are self-contained and can be reused by a number of business processes (either within or across statistical organizations). A statistical service will perform a task in the statistical process, at different levels of granularity: (i) an atomic or fine grained service may, for example, support the application of a methodological step within a GSBPM sub process, (ii) coarse grained or aggregate services may encapsulate a larger piece of functionality, for example, a whole GSBPM sub process

2. **CSPA ERROR CORRECTION: THE DESIGN**

In this section we describe two relevant pieces of the design of the error correction CSPA service, namely the CSPA Service Definition and the CSPA Service Specification.

\(^1\) ISTAT, Istituto Nazionale di Statistica
2.1. Definition

According to CSPA Specification [1], the design of a service has to be carried out by producing: (i) a conceptual-level definition of the service, where the principal business functionalities of the service are described, as well as the inputs and outputs of the service expressed according to GSIM; (ii) a logical-level specification, in which some design issues are addressed with respect to the methods of the service (e.g. how such methods will be invoked) and GSIM inputs and outputs implementation is specified according to a defined “logical” model (e.g. SDMX or DDI).

More specifically, the CSPA Error Correction Definition is described in Error! Reference source not found..

Table 1. CSPA Error Correction Specification

<table>
<thead>
<tr>
<th>Name</th>
<th>Error Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
<td>Atomic</td>
</tr>
<tr>
<td>GSBPM</td>
<td>5.3 Review, Validate &amp; Edit</td>
</tr>
<tr>
<td>Business Function</td>
<td>This Statistical Service corrects erroneous values in records.</td>
</tr>
<tr>
<td>Outcomes</td>
<td>A consistent repair of records</td>
</tr>
<tr>
<td>Restrictions</td>
<td>None</td>
</tr>
<tr>
<td>GSIM Inputs</td>
<td>Unit data set, unit data set structure (Initial raw data)</td>
</tr>
<tr>
<td></td>
<td>Unit data set, unit data set structure (Output of the CSPA Error Localization Service)</td>
</tr>
<tr>
<td></td>
<td>Rule (Edit rules)</td>
</tr>
<tr>
<td>GSIM Outputs</td>
<td>Unit data set, unit data set structure, Process Output (Type=Transformed Output) (Corrected data)</td>
</tr>
<tr>
<td></td>
<td>Unit data set, unit data set structure, Process Output (Type=Process metric) (Unsuccessful cases)</td>
</tr>
<tr>
<td>Service dependencies</td>
<td>The services is expected to be invoked after the invocation of the CSPA Error Localization Service</td>
</tr>
<tr>
<td>Process Method</td>
<td>Errorneous values that are present in records are adjusted according to the “Least Change Approach”</td>
</tr>
</tbody>
</table>

Notably: all the inputs and outputs of the service are GSIM objects, and the Process Method field of the CSPA Specification template does remark the importance of specifying the statistical method underlying the service even at the conceptual stage.

2.2. Specification

The CSPA Error Correction Specification is described in Error! Reference source not found.. It is relevant to note that: (i) the service is invoked remotely, i.e. as a Web service thus following CSPA recommendations; (ii) input and output data structures are specified according to JSON Table Schema (JTS), thus following CSPA recommendation on explicitly separate data and metadata.
3. **CSPA Error Correction: The Implementation**

Once a formal Service Specification is carried out, then you can produce several Service Implementations of it in order to provide the business function to the community.

Normally it is possible to use different ICT technologies to wrap different statistical techniques. One of the tools able to implement the Error Correction task is “*rspa*” R package and we decided to use this one although any other could have been used.

**Table 2. CSPA Error Correction Specification**

<table>
<thead>
<tr>
<th>Name</th>
<th>Error Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Protocol For Invoking the Service</strong></td>
<td>This service is invoked as a Web service.</td>
</tr>
<tr>
<td>Service location: l</td>
<td></td>
</tr>
<tr>
<td><strong>Parameters passing mode:</strong></td>
<td>All parameters are passed “by reference”.</td>
</tr>
</tbody>
</table>
| **List of parameters** | 1. Input data set  
2. Input data set structure  
3. Localization data set  
4. Structure of the localization data set  
5. Edit rules file  
6. Output corrected data set  
7. Output corrected data set structure |
| **Input Messages**    | Parameters 1) and 3) are rectangular text files following the case by variable structure with no header included.  
Parameters 2) and 4) are files JSON Table Schema compliant.  
Rule: rules are specified one by line in text format |
| **Output Messages**   | Parameter 6) is rectangular text file following the case by variable structure with no header included.  
Parameter 7) is a file JSON Table Schema compliant. |
| **Adopted methodology** | Erroneous numeric values are corrected by finding the weighted least square adjustment of values in localized fields |

In detail, we have wrapped the R script into a RestFul web service\(^2\) running on a node.js\(^3\) infrastructure, powered by the Restify library. Such a wrapper is specified by a file named *service.yaml*, in which you can find all service input and output parameters with their types and all other associated metadata, and the syntax for running the wrapped tool.

In order to invoke the service a client must make an HTTP POST request to the server, by using any kind of client-side technology such as curl\(^4\) or a web browser but also including another CSPA compliant service in a chain fashion. The content of the POST

---

\(^3\)See [nodejs.org](http://nodejs.org)  
\(^4\)See [http://curl.haxx.se/](http://curl.haxx.se/)
request consists of a JSON structure which contains all the actual paths or links to the files on which the tool will work, namely all input and output parameters.

Once the request is made, a process is spawn on the server to handle asynchronously the execution in a non-blocking manner and a job identifier is returned to the client. At any time a client can poll the status of the job by sending additional GET requests for the returned job id. When the job status is ‘finished’ the results are stored on the server and made available to the client, which can obtain them by making a GET request for each produced output resource.

Starting from the initial work done by CBS which created the overall infrastructure, we joined the development team of the existing GITHUB project adding the LEC service. It is now available at https://github.com/edwindj/cspa_rest/tree/master/LEC.

As shown in Figure 1, each NSI or organization that implements a CSPA compliant architecture can be both a service consumer or provider.

![Figure 1 - CSPA architecture and technologies](image)

4. CONCLUSIONS

The implementation of the CSPA Error Correction Service was much more than a technological exercise: (i) issues on the format/model of input and output data emerged; (ii) issues on the protocol and on the interface to implement were also raised within the Architecture Working Group (AWG) following CSPA work. The work presented in this paper shows a service compatible with the solutions provided by the AWG to such issues. More importantly, this work contributed, with the feedbacks deriving from its development, to the enhancement of the CSPA guidelines, aimed to develop better and better statistical services.

REFERENCES


5 See [http://www1.unece.org/stat/platform/display/pandp/*Home+Page](http://www1.unece.org/stat/platform/display/pandp/*Home+Page)
[2] High Level Group (HLG): High Level Group for the Modernization of Statistical production and Services, see also: http://www1.unece.org/stat/platform/display/hlgbas/High-Level+Group+for+the+Modernisation+of+Statistical+Production+and+Services