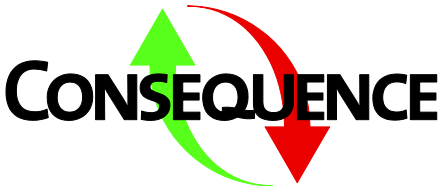


Deliverable

D1.3

# Final version of Consequence Architecture

**WP1. Architecture and Generic Implementation**

December 2010

Version 1.0

**Consequence**

*Context-aware data-centric information sharing*

FP7-ICT-2007-1

ICT-2007.1.4. Secure, dependable and trusted Infrastructures

Grant Agreement 214859

## LEGAL NOTICE

The following organizations are members of the Consequence Consortium:

Europäisches Microsoft Innovations Center GmbH,

BAE SYSTEMS (Operations) Limited,

Hewlett-Packard Italiana,

Imperial College of Science, Technology and Medicine,

The Science and Technology Facilities Council,

Consiglio Nazionale delle Ricerche,

Centre for Research and Telecommunication for Networked Communities.

**Project acronym:** Consequence

**Project full title**: *Context-aware data-centric information sharing*

---

**Work Package:** **1**

---

| | |
|---|---|
| **Document title**: | **Final version of Consequence Architecture** |
| **Version:** | **1.0** |
| **Official delivery date:** | **31 December 2010** |
| Actual publication date: | 04 January 2011 |
| **Type of document:** | Report |
| **Nature:** | Public |

---

| | |
|---|---|
| **Authors:** | Contributions from WP1 – WP6. Edited by Alexey Orlov |
| **Approved by:** | Claudio Caimi, Vaibhav Gowadia, Dmitry Starostin, Ilaria Matteucci |

| Version | Date | Sections Affected |
|---|---|---|
| 0.1 | 12.12.2010 | First draft |
| 0.2 | 20.12.2010 | First review complete |
| 1.0 | 31.12.2010 | Final review complete |

# Table of contents

# 1. Introduction

This document provides the final description of the Consequence Framework initially described in [D1.1] and [D1.2].

The Architecture presented in the document is the result of the three years of the Consequence project. Section 2 describes how the project was developing the framework throughout its lifetime.

## *1.1. The Structure of the Document*

The document does not mean to repeat the content of the [D1.1] and [D1.2] though highlights major aspect of these deliverables. The major focus of the document is the *Architecture Overview* given in Section 3. It is important to mention that this section does not contain *complete* description of all framework components but contains clear reference for such description ([D2.2], [D3.2] and [D4.2]).

Instead it focuses on describing how different components are working together implementing the process the Consequence Framework was designed for:

- Quick, dynamic and secure information sharing in multi-organisational environments with independent IT Infrastructures

The document also does not specifically describe any of the Consequence prototypes, but provides general description that may fit different scenarios. Prototype descriptions are available in [D5.4] and [D6.4]

# 2. Overview of the Consequence Approach

This section is dedicated to the overview of the Consequence project. Let us start from the general concept and objective of the project that remained the same throughout the project lifetime.

## *2.1. Consequence concept and objective*

Every organization – business, government and social – requires a steady and constant exchange of data between employees as well as with other organizations. Quite often this data may be sensitive and/or confidential but its exchange is vital for a successful organizational process. Privacy and/or business confidential requirements demand that only authorized people should be granted access to such data. Usually such authorization criteria are based upon the employee position within the organization but are not limited by it. Traditionally such *secure data exchange* was paper-based and maintained by means of controlled non-automated procedures.

### 2.1.1. Problem statement

On the market today there are many sophisticated solutions for policy based data sharing control. However there is a gap between business requirements and today's technology offerings:

- There is an increasing need for quick, dynamic and secure information sharing
- In the current scene, low-level policy systems reconfiguration is rather complicated and makes difficult their application to dynamic virtual organizations, temporary project teams and so on

- This is especially true when two or more organizations collaborate and information flows cross organizational boundaries

It would be reasonable to say that in the last case of multi-organizational domains it is often the case that the current options are for either "complete control" or "no control". In the first case, the decision may be to not share the data at all, and in the second case for completely uncontrolled sharing e.g. through email attachments. This state of affairs is clearly unsatisfactory.

Consequence intention was to take one further step and develop an architecture that will allow us to achieve effective data-sharing in dynamic environments within both single and multi-organizational domains whilst preserving a high degree of assurance on how data is handled even after it has been exchanged.

### 2.1.2. Consequence objective and expected results

The project intended to provide a *context-aware data-centric information sharing* infrastructure.

The project delivered a **Data-centric information protection framework based on data-sharing agreements.**

In particular the project intention was to:

1) Define a generic, scalable, context-aware, secure and resilient architecture within a framework to enable dynamic management policies based on agreements that ensure end-to-end secure protection of data-centric information incorporating:

- Models, algorithms, and tools for specification/authoring, elaboration, analysis, and management of multiparty data sharing agreements;
- Models and implementation of Risk and context-aware policy refinement mechanisms;
- Secure mechanisms for enforcement of controlled data sharing.

2) Engineer an interoperable software implementation of the architecture.

3) Evaluate the technical and business benefits of the implementation and framework via two test beds.

The next section describes how the project approached its major tasks.

## *2.2. Overview of the Consequence project progress*

In January 2008 the project started its first year by working on the two set of activities in parallel:

- Performing basic research for the major areas of the future Framework:
  - Data Sharing Agreements (DSA)
  - Policy and Enforcement
- Collecting the requirements and shaping basic scenarios for the project test beds that would become the basis of the Framework prototype

Within the first major activity the project performed State-of-the-Art research for each area, identified the technology to be used and defined the areas of innovation. These findings were presented in [D2.1], [D3.1] and [D4.1].

The project also presented some early demos during the First Periodic Review in February, 2009.

Finally by the end of the first year the project delivered an initial draft of the Framework architecture. The draft was based on the analysis of the test bed requirements as well as on the general understanding of the industry demands.

At the same time both test beds were busy doing extensive questionnaire within their organisations assembling business requirements for the future prototype. These requirements were then translated into the set of technical requirements and presented in [D5.1] and [D6.1].

Thus by the end of the first year the project had clear idea about the framework overall architecture, prototype requirements and technologies that will be used. The innovation work also commenced especially in the area of DSA and Policy.

The second year was dedicated mainly to development of the first framework prototype ([D5.2] and [D6.2]) that delivered core functionality of the both test beds and allowed testing the technology approaches selected by the project.

The prototype was completely implemented in the shared environment created in the cloud using Virtual Machine technology.[1]

The third and final year was dedicated to making the final version of the prototype ([D5.3] and [D6.3]) which was mainly the extension of the first one though some components were significantly improved or almost completely redone (e.g. DSA Authoring tool, DSA Analysis tool, DSA Lifecycle manager).

Throughout the first and second year the project was also pursuing related research which results did not become a part of Consequence framework prototype but nevertheless have direct relation to the project subject matter. In particular the following subjects can be mentioned:

- Modelling and analyzing of DSA using Event B language – [D2.2]
- Risk-aware Usage Decision Making in Highly Dynamic Systems – [D2.2]
- Policy Authority Evaluation Scheme (PAES) – [D3.2]
- Policy Evaluation in Adhoc Networks (Mobile PAES) – [D3.2]
- Composite Licensing for distributed Information Rights Management and Enforcement – [D4.2]
- Adaptation of the Key distribution schema for access hierarchies – [D4.2]

# 3. Overall Architecture Description

As described in Section 2 Consequence Architecture was gradually developed during the whole three years of the project going from initial ideas in Year 1 to the First Prototype during Year 2 and finally was completed during Year 3. This progress is captured in [D1.1], [D1.2] and this document.

[1] Microsoft Hyper-V

It is worth mentioning that the presented architecture does not differ a lot comparing to the one described in [D1.2] which corresponds to the project approach of gradually building the complete solution throughout the project lifetime.

The major differences between the architecture described in [D1.2] and the final one are slightly changed component interactions and introduction of *Trust Manager*. The latter is shortly described later in this document while detailed description is available in [D2.2].

## *3.1. Architecture Overview*

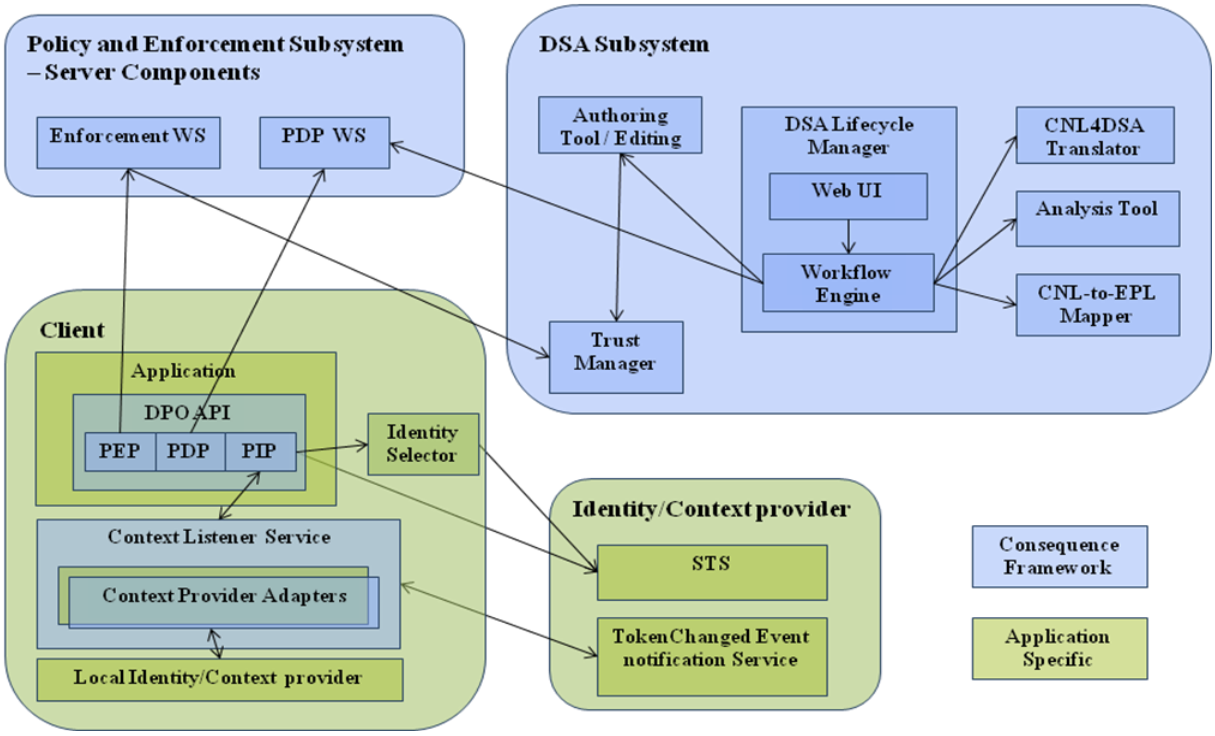


**Figure 1. Consequence Architecture**

Consequence Architecture consists of the following major subsystems (Figure 1):

- Data Sharing Agreement (DSA) Subsystem (see [D2.2])
- Policy and Enforcement Subsystem (see [D3.2] and [D4.2])

Sections below present short overviews of the major subsystems, while detailed description may be found in the correspondent deliverables (see above). Then we will illustrate the Architecture based on the complete example of Information Sharing (see section 3.2).

### 3.1.1. Data Sharing Agreement (DSA) Subsystem

Data Sharing Agreements (DSA) describe the regulations for managing shared data among multiple participants in several specific domains and contexts. The DSA is created for operations underpinning these agreements such as creation, cooperative authoring/editing, analysis, and translation from a natural language to an enforceable one. The corresponding tools for performing of these operations are provided by the DSA infrastructure.

DSA contains the following major components:

- **DSA Authoring.** This component is specifically devoted to the collaborative editing of DSA. In particular, it embodies the refinement step from natural language to high level formal specification of DSA. Strictly related to the DSA Authoring, the DSA

Vocabulary provides context-specific information. The same tool contains CNL4DSA Translator.

- **DSA Analysis.** This component is devoted to the analysis of the DSA when specified in a high level formal language.
- **DSA to Policy Mapper.** This component performs the mapping from the controlled natural language used at authoring phase to a directly enforceable policy language at system level so as to be appropriately managed by the enforcement mechanism of the Consequence framework.
- **DSA Lifecycle Manager.** This component is developed to manage all the steps involved during the DSA lifecycle, i.e., authoring, analysis, and translation to the enforceable language.
- **DSA Trust Manager.** This component deals with the distributed management of the trust relationships among principals operating on DSAs during their lifecycle.

See detailed description of DSA Subsystem and related research topics in [D2.2].

### 3.1.2. Policy and Enforcement Subsystem

The Policy and Enforcement Subsystem provides services and libraries that allow development of applications, which can be used to protect and access data under constraints defined in a Data Sharing Agreement. The Policy and Enforcement subsystem also integrates with the DSA lifecycle manager and organization specific infrastructure e.g. security token providers.

Consequence features *Enforceable Policy Language (EPL)* capable of expressing the requirements seen in the test bed scenarios. EPL policies are specified in terms of attributes of sensitive data (*metadata*) and claims (subject and contextual attributes asserted by trusted authorities). These input values are provided to policy infrastructure by the enforcement layer. Metadata is used to describe the characteristics of data, like who owns the data, how the data was collected, what the data is about, etc. In the policy infrastructure, metadata plays the role of glue that binds policies with the protected data and its users. To enable enforcement of policies across organizational boundaries, the organizations sharing data should agree on a common metadata vocabulary and associate metadata with the protected data.

While organizations need to share data, they want to control how the shared data is used after it has been given to another organization. A key requirement for controlling the usage of data is to provide *continuity* of control. This requirement is also referred as *usage control*. To control usage of data and be able to change permissions based on contextual information, a distributed policy evaluation and enforcement is needed.

Enforcement Layer Infrastructure is a set of the component and services running as locally on the client computer as well as on the remote server where server component of the Consequence framework are deployed. In the Consequence scenarios, the data have to be protected from the unauthorized usage when it leaves the boundary of a *trust domain*. Whether the data is being transferred, entering another trust-domain, or being consumed, in all these cases the Consequence framework authorizes the data usage. The enforcement layer assures that the protection applied on the data is consistent during the whole data life cycle. The enforcement-layer is designed under an assumption that sending and receiving domains cannot have absolute control over all of their communication channels or incoming and outgoing data exchanges. Instead, the enforcement layer assumes protection can be applied at point of origin when data is disseminated. Data is protected using encryption and the distribution of cryptographic keys is controlled by the enforcement layer.

The Policy and Enforcement subsystem provides following major sub-components:

1. *Policy Service* provides a central point in an organization for deploying enforceable policies and evaluating use-license requests. The DSA lifecycle manager is responsible for deploying and undeploying enforceable policies to the policy service.
2. *Client-side Policy Decision Point* (PDP) allows evaluation of individual access requests and revaluation of permissions in response to changing contextual information. We have developed extensions to core policy engine that allow it to cater for testbed requirements such as allowing access when network connectivity is intermittent and determining applicable security policies for new data derived for protected sensitive data. These extensions are described in detail in D3.2.
3. *Enforcement Service* provides services such as 1) protection of content key attached to a DSA reference, and 2) distribution of content key attached with enforceable security policies. These steps are also referred to as creation of publishing license and use-licenses.
4. *Client-side Policy Enforcement Point* (PEP) provides actual enforcement of the access decisions made by the Policy layer. The application interacts with the PEP to access data and obtain permissions. To address the Consequence test bed requirements, the enforcement layer adopts Information Rights Management model extending it to achieve the following characteristics:
    - Cryptographic protection of the disseminating data
    - Resolution of a DSA reference to domain specific enforceable policies
    - Using metadata stored with the content during content publishing (see section 3.2.3) in authorization process
    - Isolation of the enforcement client components from the publishing and consuming applications
    - Integration with different Identity/Context token providers
    - Enabling access to the protected documents in partially offline mode
    - Providing context information from the local environment
    - Performing obligations
    - Flexible authorization policies

See detailed description of Policy and Enforcement Subsystem and related research topics in [D3.2] and [D4.2].

### 3.1.3. Application Specific components

Consequence Architecture is implemented as two test bed prototypes. Each prototype contains specific client and server applications like, e.g. applications for managing research information. These applications are described in Test Bed deliverables ([D5.4] and [D6.4]).

It is important to highlight the fact that these applications may be viewed as “programmatic clients” of the Consequence Framework. In other words both prototypes are implementations of the *same* Framework but in different environments and scenarios.

Because of this reason this document does not describe specific aspects of the application dealing with the information and documents provided with the help of Consequence Framework.

## *3.2. Consequence Framework "in action"*

### 3.2.1. Overview

This section presents the way Consequence Framework works during the different phases of Information Exchange. [D5.4] and [D6.4] contain specific descriptions related to the correspondent test bed implementations. Here we will present a "general case" which may be used in many scenarios.

Overall the process of using Consequence Framework maybe divided into the following phases:

- DSA Management
- Content Publishing
- Content Usage

The sections below present description of each phase.

### 3.2.2. DSA Management

The process of DSA Management maybe divided into:

- *Establishing DSA*
- *Management of existing DSA*

#### 3.2.2.1. DSA Establishing

When two or more organizations decide to use Consequence Framework for joint Information usage, they first need to install the Framework implementations in their respective infrastructures or possibly use cloud implementations. The idea behind Consequence framework did NOT imply any connection between different organization infrastructures, except for sharing the same DSA documents and exchanging the actual data (content).

After discussing the conditions of the data sharing, actual Data Sharing Agreement is produced with the help of *DSA Authoring Tool* (see Figure 1). This tool creates an XML document featuring complete high level Information usage description in *CNL4DSA (Controlled Natural Language for Data Sharing Agreements)* developed within the project, through an English-based user editor.

New DSA document receives a unique ID and is placed into DSA Storage managed by *DSA Lifecycle Manager.* DSA Lifecycle Manager is realized as a set of applications using Microsoft Office SharePoint Server as a platform and responsible for maintaining DSA throughout its lifetime. In this case it stores the DSA and provides the opportunity for appropriate users to ignite *DSA Approval Process* which should eventually lead to acceptance of this DSA as a part of organization internal Information policies.

This approval process requires judgement from a user which seems quite logical because we are talking about business and organisational decisions. To provide help in making such decision Consequence Frameworks provides *DSA Analysis Tool* that may be invoked by DSA Lifecycle Manager. The tool performs required analysis and helps in the approval process. An example of possible analysis is the answer to questions like "Will User A get write access to a specific document under certain conditions?" Taking into account possibly lengthy and complicated structure of a DSA such analysis will help to ensure that the proposed DSA will establish Information Exchange in desired manner.

After the approval is received from user DSA Lifecycle Manager calls *CNL-to-EPL Mapper* translating DSA to *organisation specific* policies described in *Enforceable Policy Language (EPL)* developed within the project (see more details in [D3.2]). Then DSA Lifecycle calls Policy and Enforcement Subsystem and initiates *deployment* of the policies correspondent to the appropriate DSA (DSA is considered *deployed*). The appropriate policies are stored in the *Policy Storage* of Policy and Enforcement System (Policy Storage is not explicitly shown at Figure 1 though it is located as one of the components of Server Part of the system, see more details in [D3.2]).

After all participating parties (organisations) deploy the DSA users from these organisations may access the information (content) according to the rules described in the DSA.

#### *3.2.2.2.* Existing DSA Management

When participating parties decide to make changes to an existing DSA they use the process of *DSA Management.*

From the technology perspective the process is very close to described in the previous section however the business logic is different.

DSA Management may be divided in two groups of operations:

- DSA Editing
- DSA Retirement

**DSA Editing.**

When an authorized user needs to edit an existing DSA, he or she invokes the *DSA Lifecycle Manager* which provides an opportunity to select the required DSA and then invokes the *DSA Authoring Tool.* After making necessary changes the user saves a *draft* of the updated DSA. The fact that the DSA was edited is reflected in the updated version information as well as saved in the logging system of DSA Lifecycle Manager. It is important to mention that the mere editing of DSA XML Document does NOT automatically mean any changes in the deployed policies or how the content usage is being controlled.

The editing process can (and probably in real life should) be iterative and may invoke usage of *DSA Analysis Tool.*

When all necessary changes to the updated DSA are complete the user invokes *DSA Approval Process* (exactly as during DSA Establishing). Only after this process is complete and all the required approvals are received DSA Lifecycle Manager initiates creating corresponding EPL Policies with the help of the *CNL-to-EPL Mapper* (as in the case of DSA Establishing) and calls the Policy and Enforcement Subsystem to *undeploy* existing policies related to the particular DSA and *deploy* the new ones. After this process is complete the DSA Editing may be considered done.

**DSA Retirement.**

At some point in time participating parties may decide to *retire* a DSA, i.e. to stop sharing the information based upon the rules and agreements described by DSA. This can be done, e.g. due to legal data sharing agreement expiration, end of joint activity like a collaboration project, etc.

Speaking from the technology point of view DSA Retirement may be considered as a special case of DSA Editing. Like in the processes described above an authorized user invokes DSA Lifecycle Manager, selects the required DSA and initiates the process of the approval.

After successful approval the DSA Lifecycle Manager calls the Policy and Enforcement Subsystem which *undeploys* all policies related to the DSA.

DSA Retirement process should be carefully thought of while designing *collaboration business processes.* Otherwise the data shared under DSA that was later retired would become totally inaccessible, because Policy and Enforcement Subsystem will not be able to discover any policies allowing any source of access to encrypted content.

### 3.2.3. Content Publishing

By *Content Publishing* we imply a process of creating new data protected using Consequence Framework, or protecting existing data with its help.

Content Publishing may occur only when the appropriate DSA is *established* (see section 3.2.2.1) and all the corresponding policies are deployed.

Before moving any further it is important to remind, that Consequence is a *framework* that is designed to be used by different *software applications* via correspondent Application Program Interface (API). However these applications are NOT part of the framework. The project developed two test beds where the same Framework is being utilized by different sets of application used in different business scenarios. This document is not describing these specific applications. Instead it focuses on their interaction with the Framework itself and the interactions within the Framework. Complete description of the test beds may be found in [D5.1], [D5.4], [D6.1] and [D6.4].

Another important point is the notion of *Client* and *Server* part of the Framework. By *Client* components we imply the components that are needed to be *co-located* with the end-user application utilizing the Framework. *Server* parts may be (and usually are expected to be) hosted remotely.

Now, let us proceed further. When an application decides to *publish* a new content it calls *Data Protection Object Application Programming Interface (DPO API)* and creates with its help a new *Data Protection Object (DPO)* (see Figure 1). DPO is a *container* which allows many types of data formats (binary and base 64 encoded) and data access streams (object, text, memory, byte). DPO is implemented using *Open Packaging Conventions (OPC)* – a container-file technology to store a combination of XML and non-XML files that together form a single entity (see [ISO1]).

When DPO is formed it is provided with the required *metadata.* Metadata may contain a lot of information that later will be used during the content usage process (see Section 3.2.4) and the absolute minimum is the unique *DSA ID* that it used for identifying the required policies that manage complete content usage and based upon this DSA.

Also at this phase a *DPO Data Structure* is created. This contains multiple *data nodes.* These data nodes contain *references* to the actual data. Such references may be *strong* and *weak.* A *strong reference* here means that the actual data is stored within an OPC container (and thus DPO instance), while a *weak reference* provides the path (like *Universal Resource Identifier – URI*) where the actual data is stored (a record in a database system, a separate file, etc.). It is important to emphasize that in case of weak links it is the *application responsibility* to ensure that the content at the actual location is properly protected (encrypted). Consequence Framework provides all possibilities for that via the appropriate DPO API methods but does

not *enforce* the actual "remote" data protection. See [D4.1] for the complete description of DPO API.

After DPO object is formed, the application initiates the request for its protection or the actual *publishing* of content. Consequence uses *License-based Information Rights Management (IRM)* technology for Enforcing Usage policies (see [D4.1]). Such technology is based on two types of licenses – a *Publishing License* that is created during the initial protection of the content and which governs its future use and a *Use License* that is issued every time an access to the content is requested (provided that the previous Use License, if existed, has expired, see next sentence). This license governs the *actual* Use Rights for the content for a particular user within particular *context* (see Section 3.2.4) and offline-access duration specified by the license expiration time. Thus every DPO has only *one* Publishing License associated with it and then *many* Use Licenses are expected to be issued during this DPO access.

So, the DPO object is formed and the application initiates the publishing. The first step is obtaining Publishing License. *Policy Enforcement Point (PEP)* receives the request together with the associated metadata (that includes DSA ID). PEP then forwards this request to *Enforcement* component of Server Part of Consequence Framework (see Figure 1). Enforcement forms the Publishing License containing DSA ID provided with the request.

PEP then *encrypts* DPO using the content key created especially for this DPO instance. The actual encryption is done using *Certificate and Public/Private Key Infrastructure* managed by *Trust Manager* (see Section 3.2.1 and Figure 1). In case of weak data node references the application is expected to "manually" initiate remote data encryption using the appropriate DPO API methods.

After completion of the process described above the content related to the particular DPO is considered *published* and may be shared among the parties participating in the DSA.

The process of accessing the content is described in the next section.

### 3.2.4. Content Usage

By *Content Usage* we imply a process of accessing exiting content (files, data, etc.) shared and protected using Consequence Framework.

Content Sharing may occur only when the appropriate DSA is *established* (see section 3.2.2.1) and all the corresponding policies are in place.

When an application that supports Consequence Framework tries to open an object (file, data stream, etc.) that is recognized as Consequence Data Protection Object (DPO) it calls DPO API and requests access to a specific *data node.* Alternatively an application may first analyze data structure of the object and then request access to one of the data nodes revealed by this analysis.

Then Consequence Framework starts processing the content usage request. As mentioned before Consequence uses License-based IRM approach for Policy Enforcement, so for accessing the content protected with the help of this technology user needs a *Use License* (compare with *Publishing License* used during Content Publishing – see Section 3.2.3). So any case of content access will start from Use License Request.

#### 3.2.4.1. Use License Request

When application parsed the data structure of a DPO with the help of DPO API and sends an access request to the selected node, this request is received by *Policy Enforcement Point (PEP).* The request contains DSA ID that was obtained from DPO Metadata.

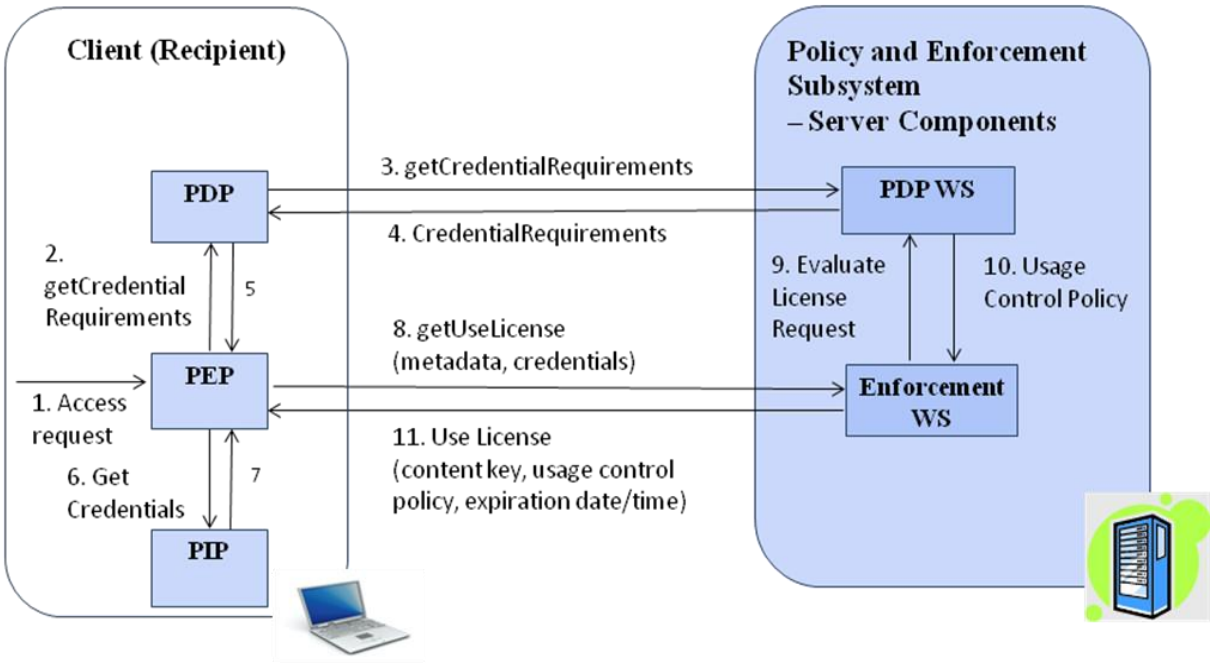


**Figure 2. Content Usage. Obtaining Use License**

PEP then forwards the request to the client part of the *Policy Decision Point (PDP)* (see Figure 2). PDP in turn queries the Server Side of PDP for credentials (tokens) needed to evaluate the access request. The PDP on the server queries Policy Storage, locates the required policy based on the associated DSA ID and determines the complete set of security tokens needed as well as Identity and Context providers that are trusted to supply them.

This information is returned to the client part of the PDP, which forwards it back to PEP. PEP then requests the *Policy Information Point (PIP)* to obtain the required security tokens. PIP contacts all appropriate *Identity and Context providers* and returns all the required information (like user ID and other context information like his/her physical location, role in the organisation, etc. – see test bed descriptions for the specific context examples).

After receiving credentials and context information (tokens), the PEP forms a Use License request containing all the required metadata, DSA ID, and credentials/tokens (including context information). The request is then sent to the *Enforcement* component of the Server Part of Consequence Framework.

The Enforcement server requests the Server part of PDP to *evaluate* the appropriate Policy (identified by DSA ID) and issues the requested Use License based on the results of this evaluation (it is important that the result of such evaluation may be "no access"; then Use License naturally will not be issued). If the requester is granted any access to the data then the corresponding content key is contained within the Use License.

The newly constructed Use License is returned to PEP. Then the Client Application can decrypt required content using the appropriate methods of DPO API. The content key for the decryption operation is extracted from the obtained Use License.

In most cases this will complete the general description of Content Usage. However in some cases user's access request is further evaluated locally to confirm that the user is allowed to perform the requested action and to control usage of the data. This process of *Usage Control* is described in the next section.

### 3.2.4.2. Usage Control

By *Usage Control* we imply the process of constant monitoring of *dynamic context information*, i.e. context information that may change in time. Good example is *physical location* of the user (implemented in Crisis Management test bed prototype – see [D5.1] and [D5.4]).

Information about requirements for such control is specified in the DSA and then translated to the policies. When client part of PDP sends the credential requests to server part of PDP (see section 3.2.4.1) it may receive a list of Context Providers that provide different *events* and how it should treat them.

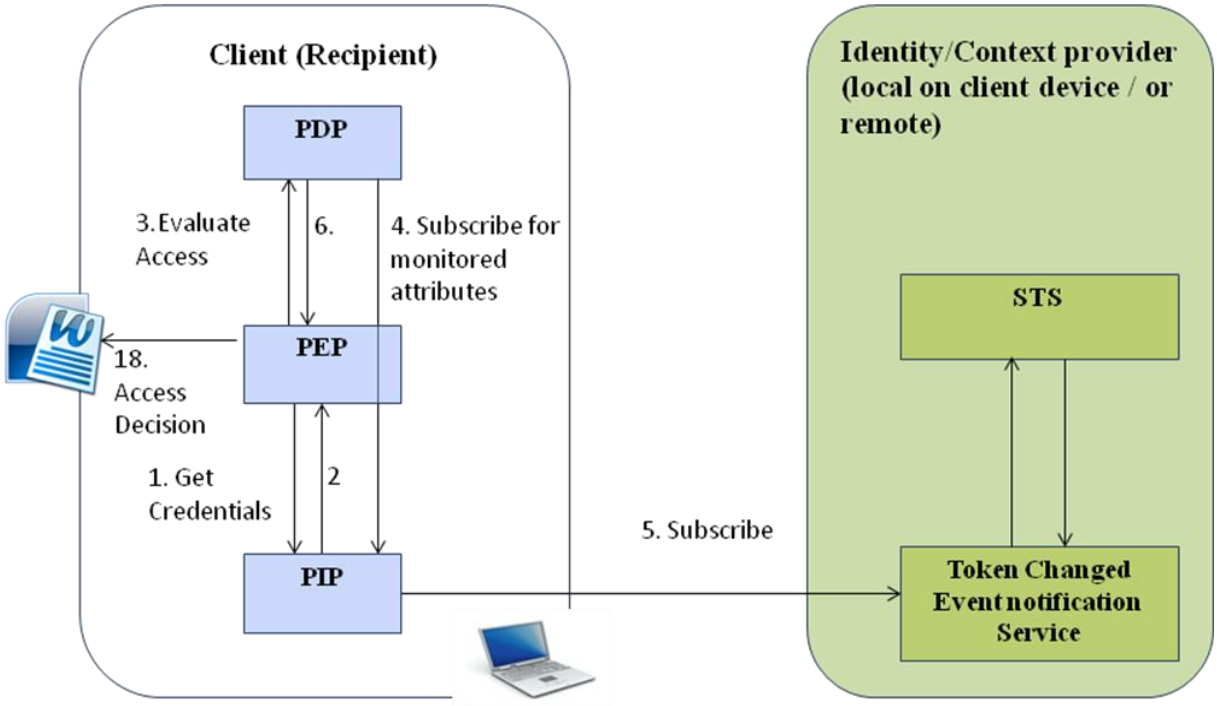


**Figure 3. Usage Control. Subscription to Context Listeners**

If such case occurs then PEP instructs PIP to *subscribe* to appropriate *Context Listeners* using *Token Changed Event Notification Service.* These dynamic Context Providers are not technically speaking a part of Consequence Framework (e.g. independent and commercially available third party product is used in Crisis Management test bed prototype for location monitoring), but needed to be integrated to the Framework via customized Context Listener Service.

The complete process of subscription is presented at Figure 3.

After all subscriptions are complete the usage control process may be considered activated. The nature of the process is very simple: every time where any of the Listeners notifies of the change (raises Token Change event) the whole access request is being re-evaluated (see Figure 4). This re-evaluation may result in Use Rights decrease or complete access denial to the formerly accessible content (as a result of e.g. the fact that the user has left secure premised and moved to a public area).

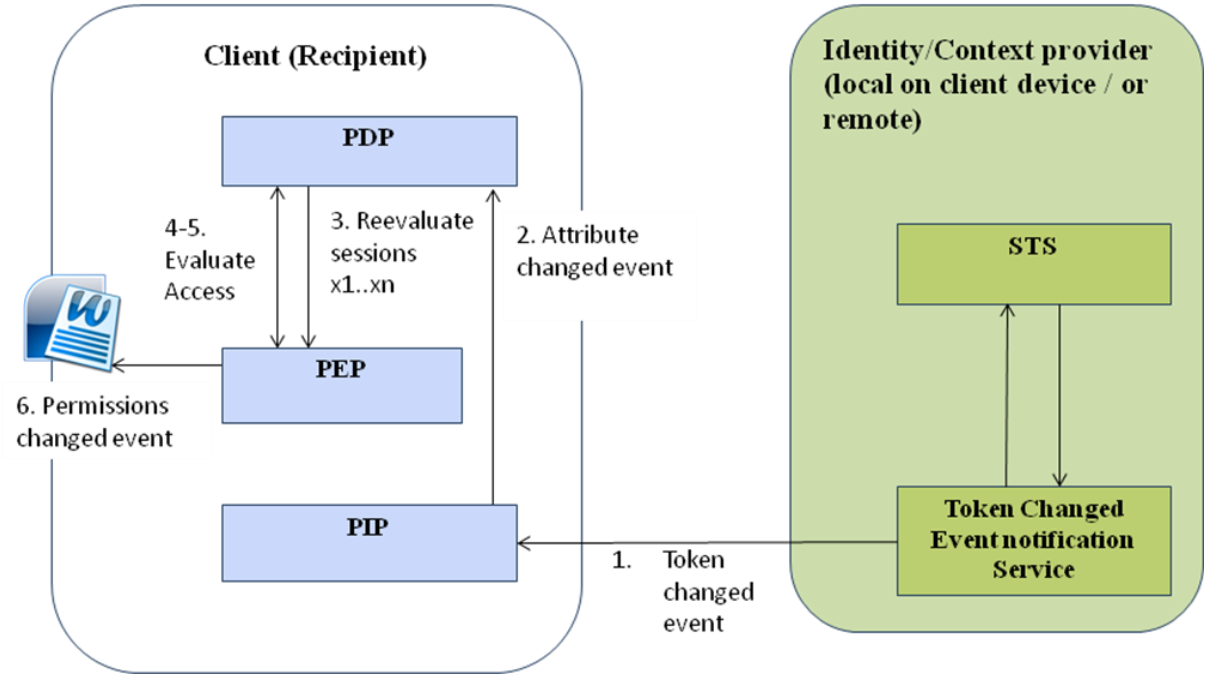


**Figure 4. Usage Control. Access Request re-evaluation after a change of Context**

# 4. Conclusion

During its three-year lifetime the Consequence Project developed and prototyped the architecture targeted to solving the problem that inspired the project:

- Improve the technology offer for quick, dynamic and secure information sharing in multi-organisational environments with independent IT Infrastructures

In solving this problem Consequence combined known technologies extending and combining them to achieve the required goal as well as delivered new research results (e.g. DSA and EPL languages).

The major effort was focused on creating and prototyping the Consequence Framework. The detailed description of Consequence prototypes may be found in [D5.1], [D5.4], [D6.1] and [D6.4].

Apart from that the Project pursued other related research topics and delivered results that did not become the part of the Framework prototype but nevertheless have direct connection with the major subject. See [D2.2], [D3.2], [D4.2] for the detailed description of this research. The same deliverables contain project opinion about possible future research areas that may be relevant to the subject and push Consequence results further.

Finally the consortium members have consistent exploitation ideas and plans to ensure that the project result will find its place in the industry.

# 5. Glossary

CNL – Controlled Natural Language

DSA – Data Sharing Agreement

EPL – Enforceable Policy Language

IRM – Information Rights Management

PDP – Policy Decision Point

PEP – Policy Enforcement Point

PIP – Policy Information Point

STS – Security Token Service

# 6. References

[D1.1] – First version of Consequence Architecture. (Project deliverable)

[D1.2] – Second version of Consequence Architecture. (Project deliverable)

[D2.1] – Methodologies and tools for data sharing agreements infrastructure. (Project deliverable)

[D2.2] – Infrastructure for data sharing agreements. (Project deliverable)

[D3.1] – Models and framework for Meta-data generation and policy infrastructure. (Project deliverable)

[D3.2] – Meta-data generation and policy infrastructure. (Project deliverable)

[D4.1] – Methodologies and tools for Enforcement Layer. (Project deliverable)

[D4.2] – Enforcement Layer Infrastructure. (Project deliverable)

[D5.1] – Consequence Requirements Specification for the Policy-Based Security for Crisis Management Test Bed. (Project deliverable)

[D5.2] – First Edition of the Policy-Based Security for Crisis Management Test Bed. (Project deliverable)

[D5.3] – Final Edition of the Policy-Based Security for Crisis Management Test Bed. (Project deliverable)

[D5.4] – Final Evaluation of the Policy-Based Security for Crisis Management Test Bed. (Project deliverable)

[D6.1] – Consequence Requirements Specification for the Sensitive Data Test Bed. (Project deliverable)

[D6.2] – First Edition of the Sensitive Data Test Bed. (Project deliverable)

[D6.3] – Final Edition of the Sensitive Data Test Bed. (Project deliverable)

[D6.4] – Final Evaluation of the Sensitive Data Test Bed. (Project deliverable)

[ISO1] – Open Packaging Conventions, International Standard ISO/IEC 29500-2:2008, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51459