SLOVENSKI STANDARD
SIST EN 16425:2014

01-oktober-2014

Vmesnik za enostavno objavljanje

Simple Publishing Interface

Schnittstelle für einfaches Publizieren

Interface de publication simple

Ta slovenski standard je istoveten z: **EN 16425:2014**

ICS:

35.240.30	Uporabniške rešitve IT v informatiki, dokumentiranju in založništvu	IT applications in information, documentation and publishing

SIST EN 16425:2014 **en,fr**

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 16425

July 2014

ICS 35.240.30; 35.240.99

English Version

Simple Publishing Interface

Interface de publication simple

Schnittstelle für einfaches Publizieren (Simple Publishing Interface - SPI)

This European Standard was approved by CEN on 22 May 2014.

CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No. EN 16425:2014 E

Contents

Page

Foreword

This document (EN 16425:2014) has been prepared by Technical Committee CEN/TC 353 “Information and Communication Technologies for Learning, Education and Training”, the secretariat of which is held by UNI.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by January 2015, and conflicting national standards shall be withdrawn at the latest by January 2015.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

This document contains the requirements for the Simple Publishing Interface (SPI), a protocol for storing educational materials in a repository.

This protocol facilitates the transfer of metadata and content from tools that produce learning materials to applications that persistently manage learning objects and metadata, but is also applicable to the publication of a wider range of digital objects.

According to the CEN/CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

1 Scope

This European Standard specifies the Simple Publishing Interface (SPI), an abstract protocol for publishing digital content and/or the metadata that describes it into repositories in a way that preserves the references between the two. This protocol is designed to facilitate the transfer of learning materials from tools that produce learning materials to applications that manage learning objects and metadata. It is also applicable to the publication of a wider range of digital objects.

The objectives behind SPI are to develop practical approaches towards interoperability between repositories for learning and applications that produce or consume educational materials. Examples of repositories for learning include educational brokers, knowledge pools, institutional repositories, streaming video servers, etc. Examples of applications that produce these educational materials are query and indexation tools, authoring tools, presentation programs, content packagers, etc.

Whilst the development of the SPI specification draws exclusively on examples from the education sector, it is recognised that the underlying requirement to publish content and metadata into repositories crosses multiple application domains.

This abstract model has been designed to be implemented using existing specifications such as v1.3 Simple Web-service Offering Repository Deposit (SWORD) profile [SWORD], Package Exchange Notification Services [PENS] and the publishing specification that was developed in the ProLearn Network of Excellence [PROLEARN SPI]. The intent of this work is thus not to create yet another specification but to create a model that can be bound to existing technologies in order to make sure that these technologies are used in a way that takes into account requirements specific to the learning domain, where it is necessary to publish both content and metadata that references it in a way that preserves these references.

The SPI model enumerates the different messages that are interchanged when publishing metadata and content.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply and are used to distinguish the requester from the system that publishes an entity (a metadata instance or a learning object):

2.1
source
system that issues a publication request. Alternatively, this system can be labelled as requester

2.2
target
system to which publication requests are sent. This can be a repository component or a middle layer component. Such a middle layer component can fulfil several tasks. It can generate and attach metadata to a resource, disaggregate and publish more granular components or act for instance as an adapter to a third party publishing API (application programming interface)

NOTE The terms “client” and “server” have not been used in order to avoid any bias towards an interface that is only applicable in client/server applications. Moreover, the scenarios in which the API is used also envisage a source running on a server (e.g., publishing from within an LMS). In the remainder of this document, the terms “resource”, “digital content”, “learning object” and “educational material” are used interchangeably.

3 Requirements and design principles

3.1 General

In this clause, some of the requirements for a publishing API are identified. These requirements stem from different repository architectures where learning resources and metadata instances need to be communicated across system boundaries. SPI enables applications to upload learning resources or metadata to a repository. For example, Figure 1 illustrates how an authoring tool (e.g., OpenOffice) could use SPI to upload a resource directly into a repository. A Learning Management System (LMS) (e.g., Moodle, Blackboard) could enable teachers to publish their materials transparently into a repository. By doing so, materials are simultaneously made available to students and published into a repository where they can be reused.

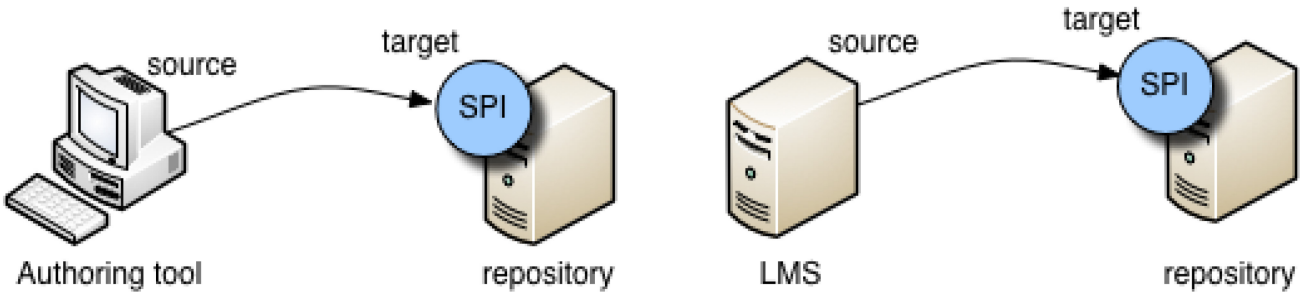

Figure 1 — Example SPI architectures

SPI also enables flexible architectures where a middleware component gathers learning resources or metadata through an SPI interface (from authoring tools or harvesters), applies value adding operations on these, and then stores them into a backend repository. Examples of such operations are disaggregation of material into small reusable components, automatic generation of metadata and validation or translation services.

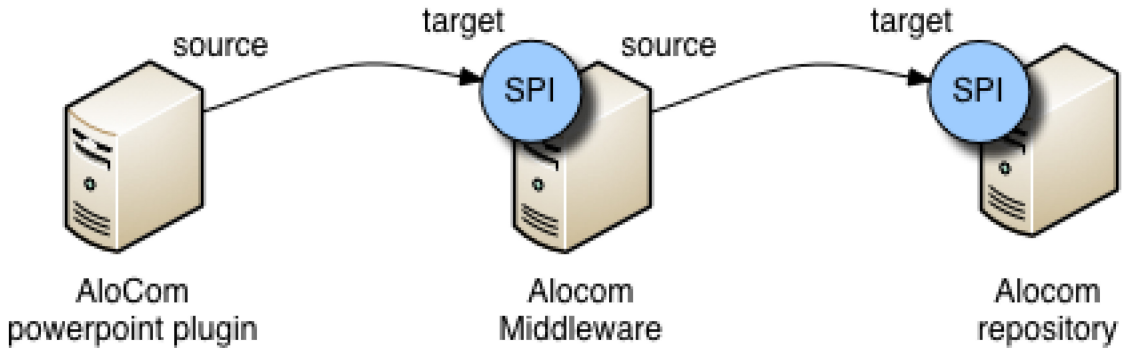

Figure 2 — AloCom architecture

Such architecture has been implemented in the context of the AloCom project (Figure 2). [ALOCOM]. This architecture contains a plug-in for MS PowerPoint, a source that can publish to a middle layer application, which is the target of this publishing operation. Next, the AloCom middleware disaggregates the material into small reusable components such as diagrams, individual slides, etc. and automatically generates metadata for each component. Each individual component is then published by the middleware component into a specialised AloCom repository where individual components are available for reuse. The AloCom middleware acts as a source and the AloCom repository as target.

Interoperability in both publishing steps is important. First, as several applications (not only MS PowerPoint) require publishing access to the middle layer application, the publishing process from within end-user

applications needs standardization. Secondly, the middle layer application shall be interoperable with other repositories, to promote interchangeability of components.

3.2 Syntactic versus semantic interoperability

The design of the SPI API is based on the design principles of the simple query interface (SQI) [SQI]. As such a simple set of commands that is extensible and flexible have been defined. By analogy with SQI, this protocol makes the following distinction between semantic and syntactic interoperability:

- syntactic interoperability is the ability of applications to deal with the structure and format of data. For instance, a language such as XML Schema Description (XSD) ensures the syntactic interoperability of XML documents as it allows for the parsing and validation of these documents;

- semantic interoperability refers to the ability of two parties to agree on the meaning of data or methods. When exchanging data, semantic interoperability is achieved when data are interpreted the same way by all the applications involved.

This European Standard tackles semantic interoperability for SPI. Without a binding (e.g., a REST binding) this specification cannot be implemented. A binding for SPI will realise syntactic interoperability.

3.3 “By reference” and “by value” publishing

Traditionally, two approaches allow for passing data from a source to a target:

- “by value” publishing embeds a learning object, after encoding, into the message that is sent to a target;

- “by reference” publishing embeds a reference (e.g., a URL) to a learning object to publish into the message that is sent to a target. This is different from publishing metadata in a referatory. Publishing in a referatory involves publishing metadata that contains a reference to the learning object. When publishing a learning object “by reference”, a reference to the learning object is used to fetch the learning object. This reference is not added to the metadata instance that describes the learning object but is used to retrieve the learning object before storing it internally.

“By value” publishing is useful for a standalone, desktop application that cannot be approached by a target in “by reference” mode. In this case, embedding a learning object in a message passed to the target lowers the threshold for pushing a learning object. “By reference” publishing is particularly suited when larger amounts of data need to be published. As embedding large files into a single message may cause degraded performance, a need exists to use a distinct method (e.g., FTP, HTTP, SCP, etc.) for transferring learning objects. Rather than imposing one of these approaches, the publish protocol will be designed to support both of them.

3.4 Flexible application

Some aspects of the SPI design follow existing applications and practices within the e-learning domain:

- a learning object referatory manages metadata that refer to learning objects stored on separate systems. Repositories that do not manage learning objects should thus be able to support SPI;

- some applications manage publishing learning objects without the metadata. For instance, PENS enabled applications submit packages to a server without metadata [PENS];

- SPI allows for publishing to repositories that manage both learning objects and metadata.

The MACE architecture for metadata enrichment [MACE] features different content providers that offer their metadata through an OAI-PMH target [OAI-PMH]. A general purpose harvester like the ARIADNE harvester is an example of a component that feeds metadata to a metadata referatory. Standardizing the publishing between the harvester and the metadata repository makes these components interchangeable.

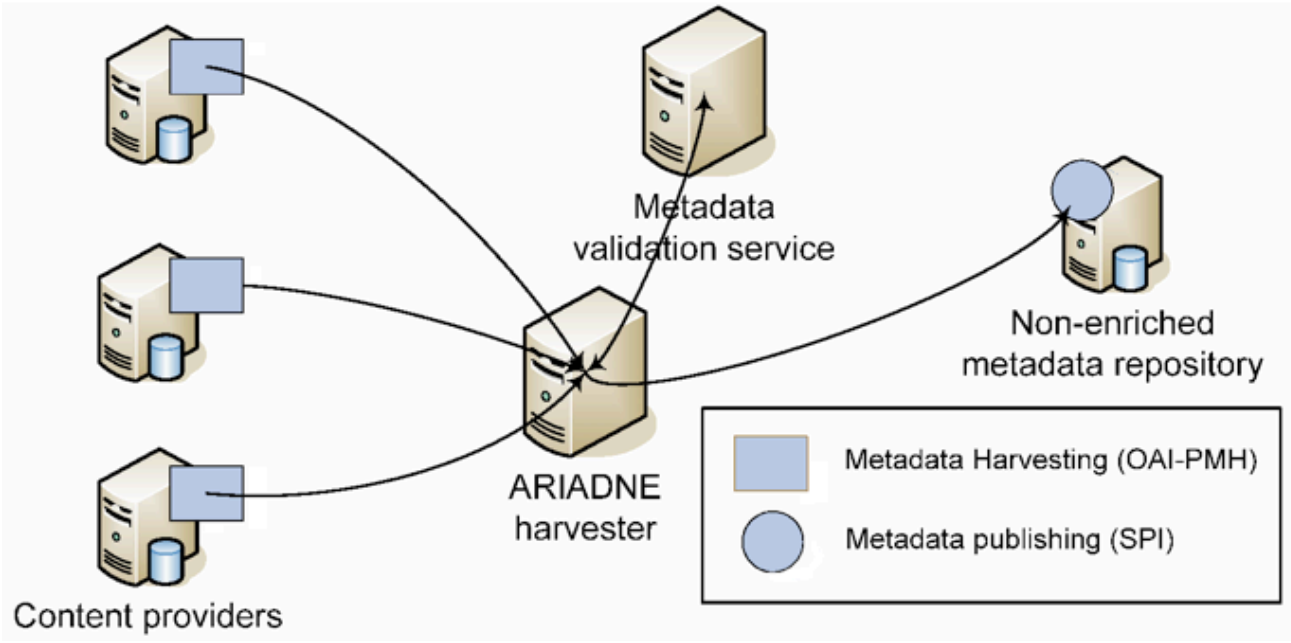

Figure 3 — The MACE harvesting architecture

3.5 Objectives

This publishing protocol meets the following objectives:

- SPI enables integrating publishing into authoring environments. This is beneficial for the author workflow, as they do not need to manually upload their learning objects using external publishing applications;
- SPI provides interoperability between applications that publish and applications that manage learning objects and metadata. Doing so, the effort of integrating publishing access into an authoring application can be reused on other learning object repositories, provided that they support SPI.

4 SPI Model

4.1 General

The model for SPI builds on a separation between data and metadata. The SPI model defines several classes of messages and functional units in a publishing architecture. When binding the specification to a given technology, these concepts are mapped into a concrete specification that can be implemented in a repository and for which conformance can be tested. All messages that are defined by the SPI model contain mandatory (M) and optional (O) elements. Mandatory means that a binding cannot relax this condition. A binding shall implement a mandatory attribute and shall make it mandatory as well. A binding can deal with optional elements in three ways. It can opt not to include the element, it can include the element and make it optional, or it can include the element and make it mandatory. A binding might for instance choose to not support transporting the filename attribute, or an SPI binding can offer support for the filename attribute while still allowing the source to provide a null value for this element. Depending on the choices made when implementing an SPI target, the latter can be configured in different ways and sources shall know the exact configuration of a target in order to be able to use it. As a consequence, the configuration of an SPI target shall be exposed to sources using at least one of the strategies presented in 4.7.

As shown by the class diagram of Figure 4, with SPI, a resource shall have an identifier (that can either be generated by a target or a source). In addition, the resource may have a filename associated. Every resource

can be described by zero, one, or more metadata instances. A metadata instance shall have a metadata identifier that identifies the metadata instance itself and shall have a resource identifier that is equal to the identifier of the resource. The metadata identifier (that can be either generated by the source or the target) enables distinguishing between multiple metadata instances referring to the same resource.

Figure 4 — Resource and metadata instance

In this model, a metadata instance shall be connected to a resource. However the resource may be hosted externally. In such a case, ingesting the resource is not part of the publishing scenario. For instance, when applying SPI to a referatory, only the messages described in 4.4 are implemented.

Alternatively, resources can be published without metadata. In this scenario, only the messages described in 4.2 are used. As an example, a single resource can be published to a repository. This scenario also includes the example of a file that consists of both data and metadata packaged in one content package.

Furthermore, this model also deals with a situation where multiple metadata instances describe the same resource.

The SPI model does not include explicit methods for updating resources or metadata instances. However, both metadata and resources can be deleted. Submitting an entity with an identifier that already exists in the target should be treated in one of the following ways by the target:

- the target overwrites the entity;
- the target creates a new version of the entity if it supports versioning;
- the target refuses to update the resource and returns an error.

Through the registry, a target can document which of the three options are supported.

4.2 Submit a resource

4.2.1 General

Submitting a resource involves sending a binary stream to a target. Depending on the binding that is used, this byte-stream can be encoded in various ways.

SPI defines two approaches for publishing a resource to a repository: “by value” and “by reference” publishing. As both methods are optional, an implementation can decide:

- to support both methods;
- to support only by reference ingesting of resources;
- to support only by value ingesting of resources; or
- not to support publishing of resources (i.e., only to support metadata publishing).

4.2.2 Resource submission by value

Figure 5 illustrates how messages are interchanged when a “submit resource” request embeds the actual resource to be published. A source first sends a message containing a resource to a target. Then, the target replies either by acknowledging a successful ingest, or by returning an error message (see 4.6).

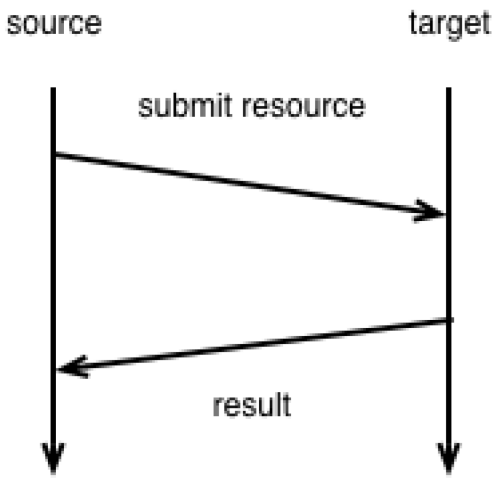

Figure 5 — By value publishing of a resource

The source can send several attributes with the message, which help the target to publish the resource. These attributes are either mandatory (M), which means that a source shall include them in the message, or might be optional (O), meaning that they can be omitted.

Table 1

Attribute	Description
Authorization token (O)	A token that enables the target to validate that the source is authorized to create a resource.
Identifier (O)	When this attribute is used, the source is responsible for generating an identifier for the resource. If this attribute is not present, a target shall generate an identifier that it returns through the result. A binding may decide not to offer support for source-generated identifiers by forbidding this attribute. When an identifier already exists in the repository, the target shall overwrite the existing resource or shall indicate that overwriting resources is not allowed or shall create a new version of the resource.
Resource (M)	The resource that will be published on the target.
Package type (O)	Identifies the kind of package (e.g., ADL SCORM, IMS Common Cartridge) that is being transmitted. A binding can for instance adopt the SWORD Content Package Types[1] to encode these. A target can use this attribute to reject the ingestion (for instance when it does not offer support for a particular package type). Furthermore, a target may use this information to unpack this package appropriately.
Content type (O)	Identifies the kind of resource that is being transmitted. A binding can for instance adopt the IANA MIME media types [IANA] to encode the

1 http://purl.org/NET/sword-types.