CERIF API specification

Version 1.0

August 2015

Editors:

Nikos Houssos, National Documentation Centre / NHRF, Greece

Dimitris Karaiskos, National Documentation Centre / NHRF, Greece

Reviewers:

Andrea Bollini, CINECA, Italy

Daniele Bailo, National Institute of Geophysics and Volcanology (INGV), Italy

Dragan Ivanović, University of Novi Sad, Serbia

Jan Dvořák, Charles University in Prague, Czech Republic

Laurent Remy, IS4RI, France

Nikos Pougounias, National Documentation Centre / NHRF, Greece

Thomas Vestdam, Elsevier, Denmark

Thorsten Hoellrigl, Thomson Reuters, Germany

Vassilis Bonis, National Documentation Centre / NHRF, Greece

Organisation names reflect the affiliations of persons at the time of their contribution to the CERIF API specification.

1 Aims and scope of the CERIF API specifications

The present document constitutes the specification of a standard API that enables 3rd party software programs to access and reuse *Research Information* maintained in Current Research Information Systems (CRIS). *Research information* can be defined as any information that describes the research output as well as the context in which research is being conducted. CRIS systems typically store, manage and disseminate data about entities like people, projects, organisations, publications, patents, products, funding programmes, indicators and metrics, research infrastructures (facilities, equipment, services) and the relationships between them. The primary objectives of the CERIF API are the following:

- To facilitate the interoperability of CRIS systems and their integration with other information systems.
- To enable the development of applications, services and components that can access and reuse information across different CRIS systems in a standard, unified way.

The CERIF API is mainly addressed to the following types of stakeholders:

- Organizations and individuals (e.g. software developers, product managers, CRIS managers) involved in the implementation of CRIS software platforms, individual CRIS systems and related services.
- Organizations and individuals (e.g. software developers, product managers) involved in the development of software applications, services and components that can benefit from retrieving and reusing the information stored in CRIS systems.

The organization responsible for the CERIF API specification is euroCRIS and in particular the CRIS Architecture and Development Task Group. EuroCRIS is the international organization for research information, a non-profit association with more than 200 institutional members from more than 40 countries in Europe and worldwide. One of the principal aims of euroCRIS is the development and curation of the Common European Research Information Format (CERIF). CERIF is the international standard data model for research information and a European Union recommendation to member states. The custodianship of CERIF has been handed over to euroCRIS by the European Union in 2000. The CERIF data structure defined as an entity-relationship model. The CERIF standard also defines an XML representation of research information (CERIF XML) as a format for data exchange involving CRIS systems. The API presented in the current document utilises CERIF XML as the basis for the representation of the data that is made available to 3rd parties by CRIS systems.

The present version 1.0 of the API addresses basic aspects of read-only access to information in CRIS systems. Operations that alter the data in CRIS systems (for example, Create, Update and Delete) as well as sophisticated search facilities are beyond the scope of this version of the API.

2 CERIF API specification

2.1 Supported operations

The calls supported by the API are documented in Table 1. The structure of responses is provided in the attached examples, as referenced from within the Table.

For the sake of brevity in the table, shortcut notation is being used for the specification within API calls of (a) paging functionality and (b) which data elements will be contained within returned CERIF Entity instances. Please refer to Section 2.2 for the relevant documentation.

#	Description	REST API request format	Example URL[1]	Method
1	Get data on all instances of a particular type of entity. The data retrieved can be (a) a list of identifiers (each identifier should be a dereferenceable URL), (b) a list of entire records. Sorting of the results according to criteria specified by the client is not supported.	*GET /{entity name in plural}?identifiersOnly=[true | false]&{pagingSpec}&{returnedEntitySpec}* ***Parameters:*** *entity name in plural [string from controlled list, mandatory]*: A CERIF entity in plural. Please refer to Section 2.3 for the list of valid values. identifiersOnly[true | false, default=true]: If **true** only an identifier (in the form of an actionable URL) is returned per record. *pagingSpec*: Specification of paged retrieval of results (see Section 2.2).	http://api.examplecris.org/projects/?offset=0&pageSize=100 Example result (attached): *identifiers.xml* http://api.examplecris.org/projects/?offset=0&pageSize=100&fedids=true&classifications=false&links=false	GET

[1] The pattern of example URLs are indicative. The current pattern assumes the availability of virtual hosting facilities, however the utilization of other patterns is allowed (e.g. http://examplecris.org/api/ instead of http://api.examplecris.org/).

#	Description	REST API request format	Example URL[1]	Method
		returnedEntitySpec: Specifies which data elements will be contained within returned CERIF Entity instances (see Section 2.2).		
2	Get all information about a particular instance of a particular type of entity. The information returned must contain the elements specified in the query. If the requested instance cannot be returned (e.g. it does not exist), an HTTP response with an appropriate 4XX code is returned.	GET /*{entity name in plural}/{id}?{returnedEntitySpec}* ***Parameters:*** *entity name in plural [string from controlled list, mandatory]*: A CERIF entity in plural. Please refer to Section 2.3 for the list of valid values. *id [string, mandatory]*: An identifier for the entity instance. Should be the same identifier as returned by API call #1 with identifiersOnly=true ("Get the identifiers of all instances of a particular type of entity") *returnedEntitySpec:* Specifies which data elements will be contained within returned CERIF Entity instances (see Section 2.2).	http://api.examplecris.org/projects/2c9083b43ec281df013ec285e81a0000?links=orgunits;fundings;persons;media *Example result*: project.xml http://api.examplecris.org/projects/2c9083b43ec281df013ec285e81a0000?fedIds=false&classifications=false *Example result*: project_short.xml http://api.examplecris.org/projects/2c9083b43ec281df013ec285e81a0000? links=orgunits&linkedObjects=true *Example result*: project_with_linked_orgunits.xml	
3	Get information about all CERIF entities (core and link entities) supported by the CRIS system (i.e. definition of the CERIF subset supported by the CRIS system).	GET /entities	http://api.examplecris.org/entities *Example result*: entities.xml	GET

#	Description	REST API request format	Example URL[1]	Method
	The returned results are structured according to CERIF-API-Entities.xsd			
4	Get semantic layer contents from a CRIS. Returns an XML dump of classes and classification schemes, such as the standard CERIF Semantics XML available at http://www.eurocris.org/Uploads/Web%20pages/CERIF-1.5/CERIF1.5_Semantics.xml	GET /semantics	http://api.examplecris.org/semantics	GET
5	Search by classification. The data retrieved can be (a) a list of identifiers (identifier should be a dereferenceable URL), (b) a list of entire records. Sorting of the results according to criteria specified by the client is not supported.	GET /*{entity name in plural}*?*class={class UUID}&classScheme={class scheme UUID}&{pagingSpec}&identifiersOnly=[true | false]&{returnedEntitySpec}* ***Parameters:*** *entity name in plural [string from controlled list, mandatory]*: A CERIF entity in plural. Please refer to Section 2.3 for the list of valid values. *class [UUID, mandatory]*: Specifies the classification term. Value: UUID of the classification. *classScheme [UUID, mandatory]*: Specifies the classification scheme to which the classification belongs. Value: UUID of the classification scheme.	http://api.examplecris.org/orgunits/?class= eda2b2ef-34c5-11e1-b86c-0800200c9a66 &classScheme= 759af939-34ae-11e1-b86c-0800200c9a66 &offset=101&pageSize=100 (class: Research Institute, class scheme: Organisation Types)	GET

#	Description	REST API request format	Example URL[1]	Method
		identifiersOnly [true | false, default=true]: If **true** only an identifier (in the form of an actionable URL) is returned per record. *returnedEntitySpec*: Specifies which data elements will be contained within returned CERIF Entity instances (see Section 2.2). *pagingSpec*: Specification of paged retrieval of results (see Section 2.2).		
6	Search by Federated Identifier. This API call is expected to return a single result. The information returned has the same form as in API call #2. If the requested instance cannot be returned (e.g. it does not exist), an HTTP response with an appropriate 4XX code is returned.	GET /*{entity name in plural}*?*fedIdClass={class UUID specifying the type of federated identifier}&fedId={value of federated identifier}&{returnedEntitySpec}* ***Parameters:*** *entity name in plural [string from controlled list, mandatory]*: A CERIF entity in plural. Please refer to Section 2.3 for the list of valid values. *fedIdClass [UUID, mandatory]*: Type (classification) of identifier. The value should specify a classification of a federated identifier (i.e. a classification belonging to the classification scheme "Identifier Types" that applies on FedId_Class). Value: UUID of the classification. *fedId [string, mandatory]*: The value of the identifier.	http://api.examplecris.org /persons/search?*fedIdClass* = 716bcc9a-c9dd-4b8b-b4ab-6c140e578ec3 &*fedId*=1234-1234-1234-1234&offset=150&pageSize=25 (fedIdClass: ORCID)	GET

#	Description	REST API request format	Example URL[1]	Method
		returnedEntitySpec: Specifies which data elements will be contained within returned CERIF Entity instances (see Section 2.3).		

Table 1. Specification of API calls

2.2 Specification of paging functionality and data elements in returned content

For the sake of brevity, the following shortcuts, as defined in Table 2, are used for the specification of API operations in Table 1 (Section 2.1).

Shortcut description	Format of query fragment	Example query fragments
Paging specification *Shortcut: (paging Spec)* How API clients specify within a request the desired paging functionality.	*offset=N&pageSize=L* ***Parameters:*** *offset [integer, optional, default=0]*: The number of items within the entire list of results that are skipped to reach the start of the current page. For example, if offset=10, the current page starts from the 11th element in the list of results. *pageSize [integer, optional, default=20, max=200]*: Maximum numbers of elements to be returned in response to the current client request. To avoid the case of servers being overloaded due to excessive page sizes, there is a maximum allowed value for pageSize (200 items). ***Note:*** No provisions are included in the API specification for the case of the result set being modified during the retrieval of the various pages (e.g. a new record is inserted possibly resulting in, for example, the inclusion of the same result in more than one pages)**.**	offset=0&pageSize=100 offset=1500&pageSize =100 offset=0&pageSize=200
Returned entity content specification *Shortcut: (returnedEntitySpec)* How API clients specify within a request which data elements will be contained within returned CERIF Entity instances.	*fedIds={true/false}&classifications={true/false}&links={true/false/{cerifEntity1;cerifEntity2;...;cerifEntityN}}&linkedObjects={true | false}&linkedSemantics={true | false}* ***Parameters:*** *fedIds [true | false, default=true]*: If **true**, the entity instance's FedIds are included in the response. Otherwise, the entity's FedIds are **not** included in the response.	http://api.examplecris.org/projects/2c9083b43ec281df013ec285e81a0000?fedIds=false http://api.examplecris.org/projects/2c9083b43ec281df013ec285e81a0000?fedIds=true&classifications=true&links=orgunits&linkedObjects=true

	classifications [true | false, default=true]: If **true**, the entity instance's Classifications ({Entity_Class} objects) are included in the response. Otherwise the entity's Classifications are **not** included in the response. *links [true | false | {cerifEntity1;cerifEntity2;...;cerifEntityN}, default=true]*: If **true**, the entity instance's links with all other entity instances are included in the response. If **false**, no links are included in the response. If a sequence of CERIF entity labels (see Section 2.3) is specified, only the links with those types of entity instances are included in the response. Labels in a sequence are separated using a semicolon (';'). *linkedObjects [true | false | {cerifEntity1;cerifEntity2;...;cerifEntityN}, default=false]*: If **true**, the entity instance's links, as full objects, with all other entity instances are included in the response, along with the entity instance at the "other" side of the link (including basic, multi-lingual and fedId fields). If **false**, full object of links are not included in the response. If a sequence of CERIF entity labels (see Section 2.3) is specified, only those links with those types of entity instances are included in the response. Labels in a sequence are separated using a semicolon (';'). *linkedSemantics [true | false, default=false]*: If **true**, the returned results include the definition (i.e. basic and multi-lingual fields) of all the semantics (classifications and classification schemes) utilized in the returned {Entity_Class} objects and links.	

	(A) Clarifications on the use of identifiersOnly. When ***identifiersOnly*** is set to **true** all other "Returned entity content" parameters are omitted. The "Returned entity content" parameters are *fedIds, classifications, links, linkedObjects and linkedSemantics*. ***(B) Clarifications on the use of links and linkedObjects parameters***: The *linkedObjects* parameter is complementary to the *links* parameter. Thus, the *linkedObjects* parameter is not taken into account when the *links* parameter is *false*. In particular: 1. If *links=false* and *linkedObjects= [true | false | {cerifEntity1;cerifEntity2;...;cerifEntity N}]* neither links nor linked objects are included in the returned results. 2. If *links=true | {cerifEntity1;cerifEntity2;...;cerifEntity N}* and *linkedObjects=false*, only the links (not linked objects) are included in the returned results. 3. If *links=cerifEntity1; cerifEntity2; ...; cerifEntityN* and *linkedObjects=true*, the specified links and the respective linked objects (i.e. only the linked objects of the entities specified in the *links* parameter) are included in the returned results. 4. If *links=true* and *linkedObjects= cerifEntity1; cerifEntity2; ...; cerifEntityN*, all links are included in the result but in full record format are only those specified by linked objects. 5. If *links=cerifEntity1; cerifEntity2; ...; cerifEntityN* and *linkedObjects= cerifEntity1; cerifEntity2; ...; cerifEntityN*, all specified links are included in the result but in full record format will be only those included in	

	both lists (the links and linked objects lists of entities). 6. If *links=cerifEntity1;cerifEntity2;...;cerifEntityN* and *linkedObjects=false*, the specified links are included in the returned results. No linked objects are included in the returned results. 7. If *links=true* and *linkedObjects=true*, all links and all linked objects are included in the returned results. ***(C) Clarifications on the use of classifications and linkedSemantics parameters***: Parameters *classifications* and *linkedSemantics* do not correlate neither they depend on each other. The ***classifications*** parameter when set to **true** includes in the response instances of type {Entity_Class} for the retrieved Entity(ies) (i.e. this parameter does not result in the inclusion of instances of type cfClass and cfClassScheme to the response). The ***linkedSemantics*** parameter when set to **true** includes in the response full records of all Class instances, i.e. Classes coming from {Entity_Class} and Classes attached on {Entity_Entity} links (e.g. the Class attached on a link of type cfPers_OrgUnit and so on).	

Table 2. Specification of paging functionality and data elements in returned content

2.3 Labels of CERIF entities

Each CERIF entity in the API is expressed using a human readable label, always in plural in any API call (e.g. projects for cfProj, publications for cfResPubl).

The entity URL labels are shown in the following Table 3. Specification of API calls:

Entity	**URL label**
cfProject	/projects
cfPerson	/persons
cfOrgUnit	/orgunits
cfResultPublication	/publications
cfResultProduct	/products
cfResultPatent	/patents
cfFunding	/fundings
cfService	/services
cfFacility	/facilities
cfEquipment	/equipments
cfMedium	/media
cfIndicator	/indicators
cfMeasurement	/measurements
cfEvent	/events
cfPAddr	/postaladdresses
cfEAddr	/electronicaddresses
cfGeoBBox	/geobboxes
cfCitation	/citations
cfCV	/cvs
cfPrize	/prizes
cfQualification	/qualifications
cfExpertiseAndSkills	/expertiseandskills

Table 3. List of valid names of CERIF entities for use in URLs

2.4 CERIF API data marshalling in HTTP

CERIF API calls are performed with the HTTP protocol, using HTTP methods. Responses to API calls return information in XML embedded in the body of HTTP response messages. The MIME type "application/xml" applies to all CERIF API responses. The returned XML data structure follows the XML Schema definitions CERIF-API-Main.xsd, CERIF-API-Header.xsd, CERIF-API-Payload.xsd.

The CRIS systems implementing the CERIF API may impose restrictions and registration / authentication requirements for clients of the API. These restrictions and requirements are beyond the scope of the current specification.

The HTTP response body contains XML divided into two parts:

1. The **header** (not to be confused with the header of the containing HTTP response message), which includes meta-information about the actual data, returned. This information, structured according to CERIF-API-Header.xsd, includes the following:
    - Data about the source CRIS system that produced this response, in particular the "base URL" of the CERIF API at this CRIS.
    - Data useful for paging results in clients (total number of returned records, actual number of records returned in current page, offset, page size, max number of records that can be returned by server in a single response). Paging results refer only to the instances of the requested type of entity as denoted by {entity name in plural} in API calls and not to the data that may be included in the response, whenever the parameters *linkedObjects* and *linkedSemantics* are used with value other than false (see Section 2.2).
    - The query that triggered this responses. This information is included only in cases of responses to GET requests and contains essentially the URL invoked by the client.
2. The **payload**, which contains the actual data retrieved from the CRIS system in response to the request. The content of the payload (which must conform to CERIF-API-Payload.xsd) can be either a CERIF XML structure or a custom XML response. The cases where a custom structure is being used are, for example, in situations where a pure CERIF XML approach would make the response overly complex and lengthy, i.e. where the gain of using a custom format is substantial in terms of complexity and size. In the current version of the API, a custom XML structure is used for the response to method call #3 (CERIF-API-Entities.xsd).

This structure is illustrated in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<CERIF-API>
    <Header>
        <api-version>1.0</api-version>
        <source>http://examplecris.org/api</source>
```

```
        <offset>50</offset>
        <pageSize>5</pageSize> <!--requested max number of elements in page -->
        <resultsInPage>3</resultsInPage> <-- actual number of elements in page -->
        <totalResults>53</totalResults>
        <maxPageSize>200</ maxPageSize>
        <query>http://examplecris.org/api/persons?offset=50&pageSize=5</query>
    </Header>
    <Payload>
        <CERIF xmlns="urn:xmlns:org:eurocris:cerif-1.6-2"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="urn:xmlns:org:eurocris:cerif-1.6-2
               http://www.eurocris.org/Uploads/Web%20pages/CERIF-1.6/CERIF_1.6_2.xsd"
               date="2014-05-04" sourceDatabase="http://examplecris.org">
         <cfPers>
               <cfPersId/>
               <cfURI>http://api.examplecris.org/persons/00581d92-dbb5-46cc-a327-68da38db9ef1
              </cfURI>
         </cfPers>
          <cfPers>
               <cfPersId/>
               <cfURI>http://api.examplecris.org/persons/03c41a9c-4b49-461c-b88f-fc1da4fabd01
              </cfURI>
         </cfPers>
          <cfPers>
               <cfPersId/>
                <cfURI>http://api.examplecris.org/persons/04f23f8b-068d-4368-a233-dbef74ecaa9f
                </cfURI>
         </cfPers>
        </CERIF>
    </Payload>
</CERIF-API>
```

Appendix: CERIF API design choices

A range of design issues is encountered during the definition of the CERIF API. The following Table 4 lists the main issues involved and mentions, point by point, the approach followed based on the discussions and feedback within the Arch TG.

Issue	Approach
Which API technology and protocol should be used, SOAP or REST?	REST, due to its simplicity for developers, ubiquity and inherent support for the operations foreseen for the CERIF API (i.e. generic, fundamental CRUD-style primitives – at first limited to Read – not complex "business" operations).
What should be the output format technology, XML or JSON?	XML, since CERIF XML is already in place. JSON has been discussed as an idea (Porto joint CERIF and Arch TG meeting), but has not gained wide acceptance within the euroCRIS community at the moment.
The data returned by API calls will be always strictly CERIF XML?	Having everything as CERIF XML is compatible with current CRIS systems, since any data the server and client need to produce and parse is the CERIF XML Schema. However, using CERIF XML for everything adds considerably to the complexity and size of the exchanged data, especially for the /entities and /{entity name in plural} calls (e.g. lists of identifiers or counts can be transferred more efficiently using a custom encoding instead of representing them in CERIF XML, e.g. as cfURIs and cfMeasurements). While in CERIF XML the size of the data exchanged through these API calls is not expected to be extremely large in most cases, it is not unlikely that the volume of exchanged information might impact performance in certain situations, while simplicity in representation will be beneficial for API implementors. Furthermore, some meta-information about the returned results is useful to be included in API responses. Therefore, the adopted approach is to represent the actual returned CRIS data in most cases in CERIF XML, with certain exceptions when an alternative representation is substantially simpler and less verbose. Furthermore, a header section is included with every API response, providing meta information about the returned results in a new non-CERIF XML format.

REST is an architectural paradigm with a wide range of implementations and "RESTFul-ness" is a debated topic. How strict will the CERIF API be with conforming to the "orthodox" REST way of creating an API? A common distinction is the levels of a REST maturity model defined by Richardson et al[2]: Level 0: HTTP as a transport mechanism Level 1: Model data as resources addressable by URIs Level 2: Use HTTP as an application protocol (i.e. HTTP verbs for defining operations, HTTP message codes for addressing exceptions) Level 3: HATEOAS (Hypertext As The Engine Of Application State)	Level 2 has been followed in the current version of the API specifications, with some features of Level 3. Level 3 might be useful, but also probably to some extent an overkill for the case of the CERIF API and are not yet widely used by developers in real-life APIs. A feature of Level 3 has been (partly) adopted in the current draft API specifications: Certain types of responses contain hyperlinks that the client can follow to continue retrieving and consuming data from the server through the API. For example, each entity in the /entities response contains a link to the list of entity instances for this entity and the latter (list of entity instances for an entity) contains links to each entity instance of this type – all links are represented in valid CERIF XML.
What content / MIME type should be used in HTTP interactions between client and server? Is application/xml enough or should a custom media type be defined?	This is a probably useful feature, but mostly important when HATEOAS is being used. In the current version **application/xml** is being used.
"Depth" of information retrieved with each entity instance.	A purist REST approach would be to separately retrieve records of different entity types (e.g. retrieve a publication and its authors through two separate requests). In practice, it is convenient for developers, in particular especially CERIF API consumers, to be able to retrieve a selected number of entity instances linked with a CERIF XML record (e.g. retrieve authors and projects linked with a publication record). Therefore, the CERIF API foresees such functionality in a parameterized way, so that retrieval of information is selective (to avoid fetching unnecessary records) and applied only in cases that is meaningful for the client (see Section 2.1, the *linkedObjects* parameter in API call #2)

Table 4. CERIF API design choices

[2] Richardson, L. (2008) Justice Will Take Us Millions Of Intricate Moves. Presentation at the QCon 2008 conference, 19-21 November 2008, San Francisco, USA. Available at http://www.crummy.com/writing/speaking/2008-QCon/.