EDSSI

IIA V6 Exchange and Signing Behaviour

Document title	**European Digital Student Service Infrastructure** **IIA V6 Exchange and Signing Behaviour**
Version	**1**
Status	**Published**
Author(s)	**EDSSI**

IIA Partners Approval Behaviour

Assumptions

Partner A has all needed IIA APIs in the latest version

- IIA Get API
- IIA Index API
- IIA CNR API
- IIA Approval CNR API
- IIA Approval API

Partner B has all needed IIA APIs in the latest version

- IIA Get API
- IIA Index API
- IIA CNR API
- IIA Approval CNR API
- IIA Approval API

Exchange and Signature Behaviour

1. Partner A creates an IIA in their system
   1.1. Upon this creation, or at a later stage, but before exposing the IIA to the EWP network, Partner A can record their signing date of this IIA, giving Partner B the date of its signature, if signed.
2. **Partner A** should contact the **IIA CNR** of **Partner B** and receive a successful answer, meaning that the CNR has been successfully received
3. **Partner B** does a request to **Partner A IIA Get API** , either after receiving the CNR or at any moment in time
   3.1.1. When **Partner A** is generating the IIA XML should **calculate the Hash of the Cooperation Conditions** and append it in the XML
   3.1.2. When **Partner B** retrieves the XML can also **calculate the Hash of the Cooperation Conditions** received and **compare** to the Hash received (appended in 3.1.1)
   3.1.2.1. If they match, it means that the information has not been altered since the hash has been calculated on the partner side.
   3.1.2.2. If they do not match, something may be wrong and conditions may have been altered since the hash has been calculated on the partner side.

If, at this moment, **Partner B** decides to **edit the IIA** in their system, the process should start all over, from **Point 1**, with inverted names.

4. **Partner B** is happy with the IIA received by **Partner A**
   4.1. Partner B internally signs the IIA and records their signing date of this IIA
   4.2. **Partner B** puts **Partner A Hash** listed in its **IIA Approval API**
   4.3. **Partner B** contacts the **Partner A IIA Approval CNR API** and receives a successful answer, meaning that the CNR has been successfully received
   4.4. Partner A contacts **Partner B IIA Approval API** checking if their hash for the IIA ID received in the IIA Approval CNR API is present there, meaning that **Partner B** has approved **Partner A** version of a specific IIA

At this moment, Partner A has its copy of the IIA **signed by both parties but only approved by Partner B**, by exposing Partner A's Hash in Partner B's IIA Approval API. After this, the following scenarios are accepted:

- Both partners **agree that having the signature date from both sides is enough** to complete the IIA and end the IIA data exchange
    - In this case there are no more steps to follow
- **Partner B also wants his copy of the same IIA to be Approved** by the partner
    - Steps from 2 to 4 should be followed changing partners A and B
    - Partners A and B may have different copies of the same IIA data so there may be, or not, a direct connection between the IIA ID of Partner A and Partner B
- **Partner A and B may have different IIAs for the same purpose**, for example, Partner A may have an IIA which covers several IIAs of Partner B
    - For each of these agreements, steps 2 to 4 should be respected

Additional remarks

- The signing section is not part of the hash calculation.
- Each HEI fills its own signing section of the IIA.
- If HEI gets the partner's version of the IIA with the signing data of the partner, it may copy it to the local version of IIA.
- It's up to HEI to decide if it will approve the IIA version of the partner when the signing section of the partner is empty. It makes sense to approve IIA only when the signing section is filled.
- It's up to HEI to decide if it wants to get its own version of the agreement approved by the partner.
- HEI can not refuse the approval of the partner's version of the agreement with no good reason.
- It's up to HEI to decide when to set the **in-effect** flag to TRUE. It makes sense to set it to TRUE after getting the partner's approval of the local version (if the partner has approved my version I can start nominating students etc.)
- Approvals can be done with\without both signing date and **in-effect**.

IIA Hash Calculation

Calculation of the Hash when generating the XML

When receiving a request, to it's **IIA GET API** from **Partner A**, **Partner B** has to generate an XML document with all the information, including the Cooperation Conditions of that specific agreement, and has to **calculate the Hash of these Cooperation Conditions**, appending it to the XML document, on its specific tag.
For this calculation, **Partner B** has to take into account the following points:

- The hash **should not** change if the cooperation conditions are not really changing.
- The `sending-contact` and `receiving-contact` subelements are **not** taken into account when calculating hash.
- Before calculating the hash, the cooperation-conditions element should be normalized using Exclusive XML Canonicalization.

Hash calculation pseudocode may be found in the "Hash calculation" section.

Calculation of the Hash when receiving an XML

Upon making a request to a Partner B IIA Get API, if successful, Partner A receives an XML document that, if valid (please validate in the Schema Validator[1]), will contain a Hash tag that was calculated by Partner B, focusing on the Cooperation Conditions, during the generation of the document.
For this calculation, **Partner B** has to take into account the following points:

- The `sending-contact` and `receiving-contact` subelements are **not** taken into account when calculating hash.
- Before calculating the hash, the cooperation-conditions element should be normalized using Exclusive XML Canonicalization.

After the calculation, **Partner A** will have both its **calculated hash** and **Partner B hash**. At this point Parner A can check if the hash of Partner B was correctly calculated and that Cooperation Conditions haven't been altered since last request.

Hash calculation pseudocode may be found in the "Hash calculation" section.

Hash calculation

There is an important consideration to the Exclusive XML Canonicalization. The cooperation-conditions element has to contain the same namespace aliases as the XML response to the IIA GET method. If a namespace alias is autogenerated when marshalling to XML, then it might be a good idea to set the namespace alias to a predefined value.

An a example pseudocode for hash calculation:

- Extract cooperation conditions from the IIA GET response object.
- Remove sending and receiving contacts elements from the extracted object.
- Apply Exclusive XML Canonicalization (take into consideration namespace aliases as mentioned above!).
- Hash the calculated XML with SHA256.
- Add to the IIA GET response a conditions-hash element with the calculated hash.

[1] https://dev-registry.erasmuswithoutpaper.eu/schemaValidator

Useful Links

https://en.wikipedia.org/wiki/Canonical_XML
https://www.w3.org/TR/xml-exc-c14n/

www.edssi.eu