ADUCID Advanced Integration Manual

Version 3.1.0.RC3

Release date June 17, 2016

Antala Staška 2027/79, 140 00 Prague 4 | T +420 271 100 100 | F +420 271 100 101
Vídeňská 125, 619 00 Brno | T +420 547 100 100 | F +420 547 100 101
Sady 5. května 59, 301 00 Pilsen | T +420 271 100 100 | F +420 271 100 101
Teslova 30, 821 02 Bratislava | T +421 (2) 4821 3111 | F +421 (2) 4821 3199

ISO 9001 ISO 14001 ISO/IEC 27001 ISO/IEC 20000
CZ > Corp. ID 25313029 | Registration Court RC Brno, File no. B2113
SK > Corp. ID 35787546 | Commercial Register DC Bratislava, Section Sa file 2431/B

Table of Contents

1. Purpose of this document

This document serves as an integration manual for incorporating ADUCID® technology into third-party web applications. Two methods of integration are described in the document:

- Integration using the AIM-Proxy component
- Integration without using the AIM-Proxy component

Both integration methods are based on HTTP redirect.

2. Prerequisites

For understanding of this document, knowledge of the following documents is required:

- aducid-architecture.pdf

Knowledge of web technology, programming and integration of web applications is required.

3. Principles of ADUCID integration

Fundamental components and the way they communicate with each other, including basic principles of integration, are described in the document named "ADUCID Architecture". For easy orientation, the components are summarized below from the integrator's point of view.

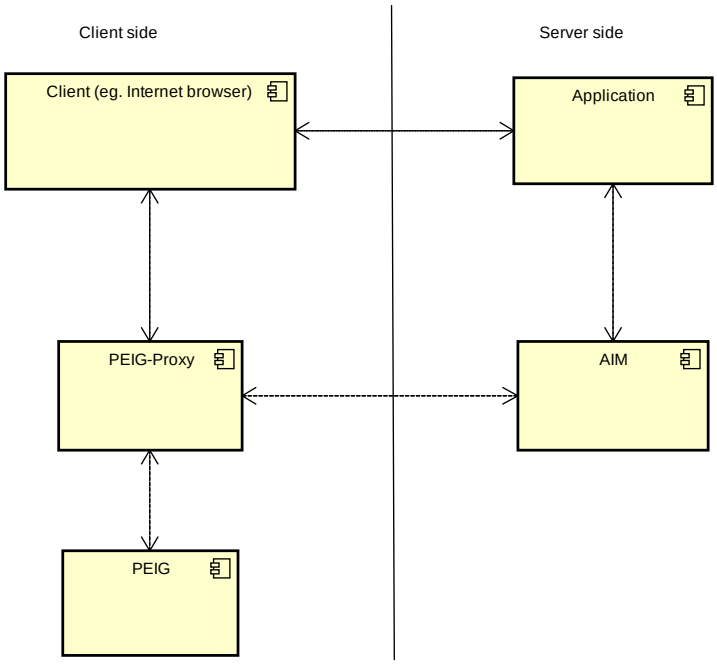

Figure 3-1 ADUCID components

- **Application** - A component that integrates ADUCID technology. The application communicates with AIM during authentication.
- **AIM (ADUCID Identity Machine)** - A self-authenticating component. Delivers the interface for working with the identity and the authentication protocol interface.
- **PEIG-Proxy** - A client-side component that is used for communication between AIM, PEIG and the web browser (other client).
- **PEIG** - A component that maintains identities and performs authentication. It communicates with AIM through the PEIG-Proxy.

• **Client** - Usually an Internet browser.

3.1. Authentication "user story"

This chapter defines the typical authentication workflow from integrating application's perspective:

1/ The application initiates an authentication session in AIM.
2/ The integrator forces the initiation of the authentication process on PEIG (it uses either the AIM-Proxy component or its own resources).
3/ Authentication is performed between PEIG and AIM, and the credentials are sent back to the integrating application.
4/ If the verification is successful, the application fetches user profile data from AIM.

The following text describes two ways to implement this scenario which demonstrate the basic integration principles:

• Use of the AIM-Proxy server adapter (simpler use)
• Direct control of the client part (transmission of requests to PEIG needs to be programmed)

3.2. Use of the AIM-Proxy server adapter

AIM-Proxy is supplied to simplify the authentication process through HTTP redirect — on PC platform over an HTTP redirect adapter or on a mobile platform over a direct PEIG-Proxy call. The application can delegate the authentication process to these components. AIM-Proxy for integrators delivers the IAIM-Proxy interface.

The figure on the next page illustrates how AIM-Proxy is used during the ADUCID authentication process.

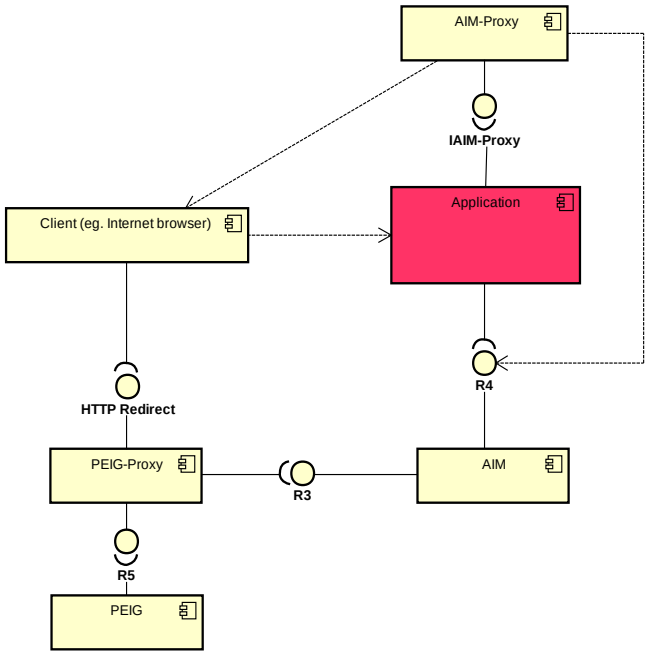

Figure 3-2 Integration using AIM-Proxy

The following sequence diagram illustrates individual authentication steps.

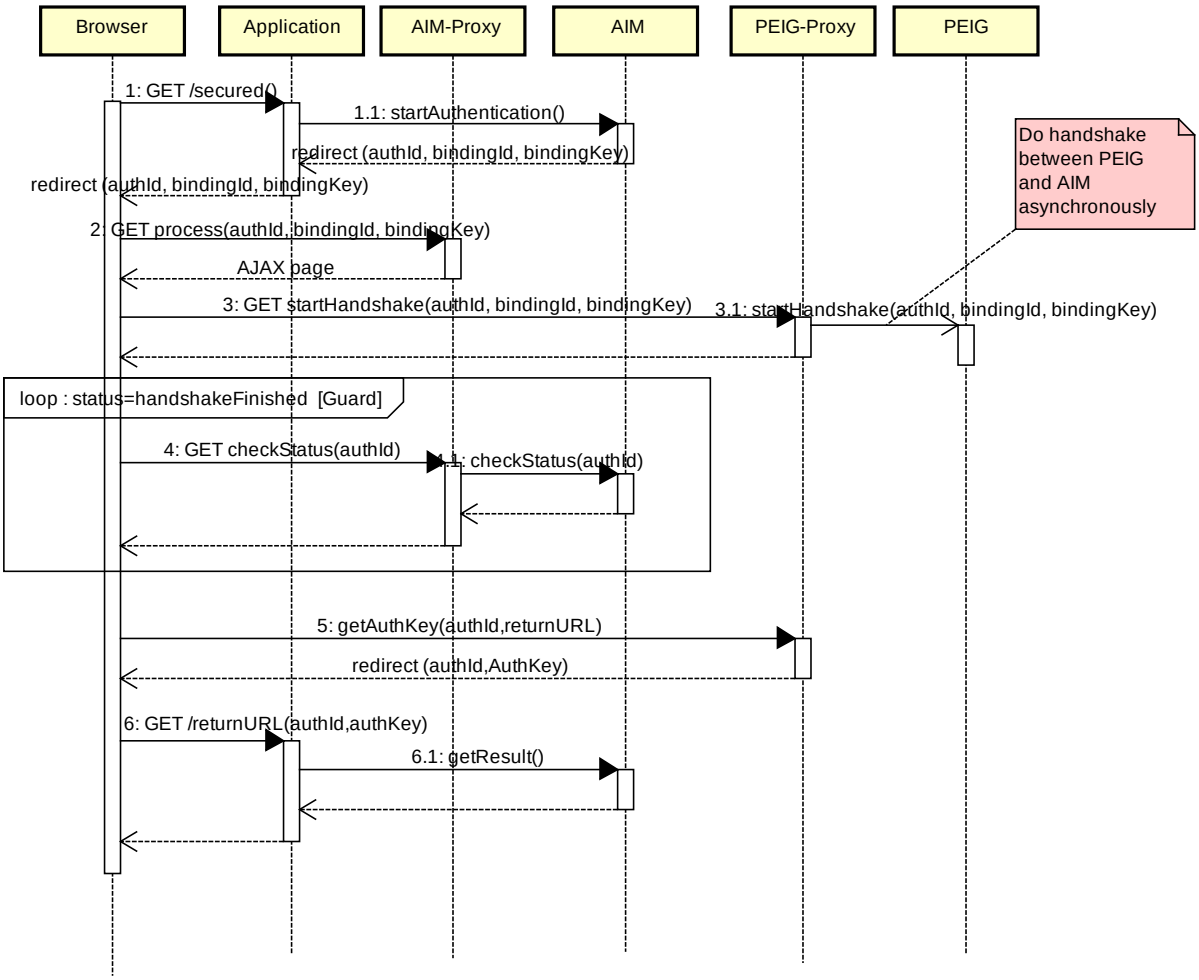

Figure 3-3 Sequence of events when using AIM-Proxy

1/ The user moves to the secure page of the given application.

2/ The application creates a new authentication session in AIM (it uses either ADUCID Java SDK or direct access to AIM via the R4 interface). This generates an **authId** and the optional **bindingId** and/or **bindingKey** identifiers for the authentication session. These parameters are then transmitted to the AIM-Proxy component via HTTP.
*Note: **authId**, **bindingId** and **bindingKey** must be converted to Base64. All parameters must be escaped prior to calling HTTP redirect URL.*

3/ The AIM-Proxy component generates an AJAX page, which performs the following actions:
   a/ It initiates the client-side authentication process, it is done through the following steps:
      i/ Checks, if the ADUCID schema is supported, if it is, it calls this schema — **this causes the start of mobile phone PEIG**
      ii/ If step i/ was not successful, it tries to push to the HTTP redirect adapter — **this causes the authentication session to start over PC PEIG, if it is running**
      iii/ Without dependency on the success of steps i/ or ii/ **the QR code is prepared and shown to the user**, if it is supported by current binding mode, The QR code is an alternative way to start an authentication session
   b/ It periodically checks the progress status of authentication against AIM:
      i/ On a mobile phone platform, PEIG-Proxy checks the authentication status
      ii/ On a PC platform, AJAX page checks the authentication status
   c/ If the AJAX script or a mobile phone PEIG-Proxy detects a successful authentication, a query is sent from the HTTP Redirect adapter or a mobile phone PEIG-Proxy component to obtain the authKey.

d/ HTTP Redirect adapter or a mobile phone PEIG-Proxy redirects back to the application (returnUrl), where adding the authId and authKey to the URL to verify and complete the authentication process.

4/ The application verifies the authId and the authKey against AIM.

3.2.1. AIM-Proxy Interface (IAIM-Proxy)

The interface of the AIM-Proxy component is based on HTTP protocol and defines the following commands (the interface address is `http://<aim_host>/AIM-proxy/{operation_name}`):

Command Name	Description
/process	This command returns the AJAX script, which facilitates authentication. In most cases, this command is called by the HTTP redirect with the following parameters: authId — Mandatory, **URLEncoded** and **Base64Encoded** identifier of the authentication session (acquired via ADUCID Java SDK or direct access to the R4 interface). bindingId — Mandatory, if provided during the session start, the **URLEncoded** and **Base64Encoded** identifiers are required for client binding (acquired via the ADUCID Java SDK or direct access to the R4 interface). bindingKey — Mandatory, if provided during the session start, the **URLEncoded** and **Base64Encoded** keys are required for client binding (acquired via the ADUCID Java SDK or direct access to the R4 interface).
/checkStatus	A command for fetching the ongoing status of authentication. URL parameters: authId — Mandatory, **URLEncoded** and **Base64Encoded** identifier of the authentication session The command returns the authentication status as defined in chapter 4
/processReturnUrl	A command for fetching the **returnUrl** — URL, where to return to after a successful authentication. This URL is an input to the authentication session start operation, see the ADUCID Java SDK for more information. URL parameters: authId — Mandatory, **URLEncoded** and the **Base64Encoded** identifier of the authentication session
/version	Fetches the AIM-Proxy version. The command returns the version number in plain text format.
/qrCode	This command generates an image representing the QR code as an authentication start alternative. This image can then be shown via a web interface. When this image is taken by a mobile phone, the authentication session automatically starts. URL parameters: authId — Mandatory, **URLEncoded** and the **Base64Encoded** identifier of the authentication session (acquired via the ADUCID Java SDK or by direct access to the R4 interface). bindingId — Mandatory, if provided during the session start, The **URLEncoded** and the **Base64Encoded** identifiers are required for client binding (acquired via the ADUCID Java SDK or by direct access to the R4 interface). bindingKey — Mandatory, if provided during session start, The **URLEncoded** and the **Base64Encoded** keys are required for client binding (acquired via the ADUCID Java SDK or by direct access to the R4 interface).

Example of using the interface:

```
http://<aim_host>/AIM-
proxy/process?authId=%2fVRUFhn8ISY%3d&bindingId=1AB5HQxtZJ0%3d&bindingKey=zrlnWxxhCFg%3d
```

3.3. Direct control of the client part

This scenario represents the situation where the integrating application starts the authentication process in its own manner. Typically, this means inserting an AJAX script, which the AIM-Proxy component produces, into the application, so that the HTTP requests do not leave the context of the application, thus increasing the application's security.

The integration is schematically illustrated in the following figure:

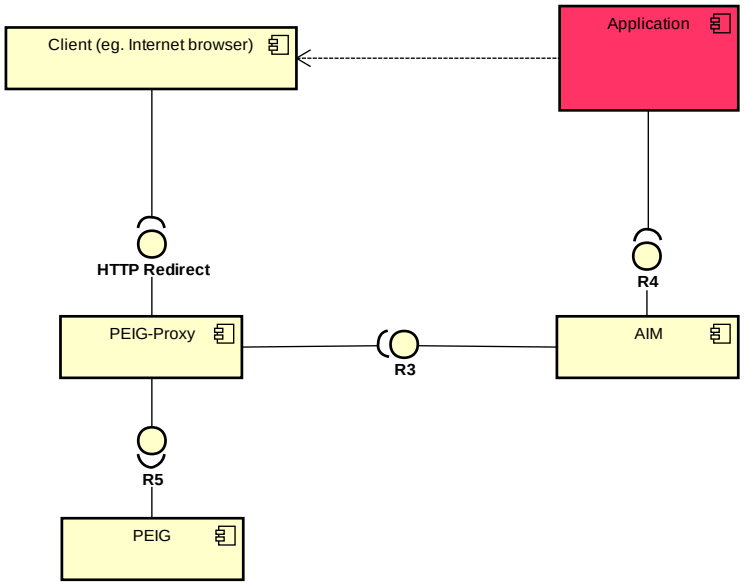

Figure 3-4 Integration with direct control of the client part

If the integrators want to develop their own way of interacting with PEIG, they can use the HTTP Redirect interface of the PEIG-Proxy component on a PC platform or by using the PEIG-Proxy on a mobile phone platform, which are defined as below.

3.3.1. HTTP Redirect interface of the PEIG-Proxy component — PC platform

This interface is typically used directly by the web browser through an AJAX script.

The interface listens at the 44240 port of the user's computer and facilitates communication between the web browser and PEIG-Proxy using the HTTP protocol.

Two endpoints are available for the integrator with different input parameters:

Endpoint	Description
The endpoint used to run the authentication process (`http://localhost:44240/ADUCID/PEIG/auth`)	This command returns a true/false value in plain text format. In case of fatal error in the endpoint, the HTTP 500 status is returned. URL parameters: id - base64 URL of the encoded **authId** bindingId—base64 URL of the encoded **bindingId** bindingKey—base64 URL of the encoded **bindingKey** url - AIM URL, against which the authentication is performed (URL of the R3 interface)

Endpoint	Description
Endpoint used to complete the authentication on the client and fetching the **authKey** (`http://localhost:44240/ADUCID/PEIG/auth`)	This command can be called if authentication has finished (by checking against the AIM server, e.g. via the checkStatus command) The command results in an HTTP redirect to URL in the application where the credentials will be verified. The redirect address includes the authId and authKey parameters. URL parameters: authId - base64 URL of the encoded authId

Example of running the authentication process:

```
http://localhost:44240/ADUCID/PEIG/auth?id=%2fVRUFhn8ISY%3d&bindingId=1AB5HQxtZJ0%3d&bindi
ngKey=zrlnWxxhCFg%3d&url=http%3a%2f%2flocalhost%3a8080%2fAIM%2fservices%2fR3
```

Example of completing the authentication process on the client side:

```
http://localhost:44240/ADUCID/PEIG/auth?authId=%2fVRUFhn8ISY%3d
```

3.3.2. PEIG-Proxy — mobile phone platform

All mobile phone platform applications are called over schema. This endpoint is defined to call a mobile phone PEIG:

Endpoint	Description
The endpoint used to run the authentication process (`aducid://callback?...`)	This command calls a mobile phone platform PEIG, if one is installed URL parameters: authId—base64 URL of the encoded **authId** bindingId—base64 URL of the encoded **bindingId** bindingKey—base64 URL of the encoded **bindingKey** r3url—AIM URL, against which the authentication is performed (URL of the R3 interface)

The following is an example of starting the authentication process on the mobile phone client side:

```
aducid://callback?authId=%2fVRUFhn8ISY%3d&bindingId=1AB5HQxtZJ0%3d&bindingKey=
zrlnWxxhCFg%3d&r3Url=http%3a%2f%2flocalhost%3a8080%2fAIM%2fservices%2fR3
```

After successful authentication and fetching **authKey**, the mobile phone PEIG returns control back to the browser that started the authentication process.

3.4. AIM (R4) Interface

The AIM server provides the integrator with an R4 interface, which is accessible only to applications on the server side.

The communication interface is based on SOAP/HTTP and consists of a single R4 service available at:

```
http://<aim_host>/AIM/services/R4
```

This interface is designed to facilitate direct communication of server applications with the authentication server and offers 4 basic commands:

Name of operation	Description
AIMrequestOperation	Command for work with electronic identity and pocket personal objects (PPO).
AIMgetPSLAttributes	Command for fetching the results of operations performed with electronic identity.
AIMexecutePersonalObject	Command for work with directory personal objects (DPO).
AIMcloseSession	Command for program termination of an authentication session.

Table 1: Basic commands of the R4 interface

A WSDL describing the R4 interface is included on the supplied DVD in the following directory: integration/wsdl-xsd.

For easier work with the R4 interface, a Java SDK library has been created, which is described in a separate document, aducid-sdk-java.docx. To simplify the use of the R4 interface, we recommend using the supplied SDK, or reading the provided document.

The ADUCID Java SDK source codes distributed under Apache Server Licence 2.0 provide information on how to specify individual attributes of the web service.

4. Detailed view of ADUCID authentication

The authentication process typically consists of the following steps:

1/ The application obtains a unique identifier **authId** and the optional **bindingId** and/or **bindingKey** (AIM typically has it generated by the authentication server using the startAuthenticationSession command in SDK).
2/ The application ensures the transmission of **authId** and the optional **bindingId** and/or **bindingKey** and AIM URL to the PEIG-Proxy component (either by its own means - e.g. its own AJAX script - or by using the AIM-Proxy component).
3/ PEIG performs an authentication handshake with the AIM authentication server via PEIG-Proxy (transmitted as an input URL parameter to the PEIG-Proxy component).
4/ The secret (**authKey**) generated is then returned to the application at the specified URL where the credentials are verified.
5/ If the credentials are successfully verified, the application can work with the results of the command (with authentication session) for a predefined time period.

The integrator working with the R4 interface can obtain information on the current status of the authentication using the AIMGetPSLAttributes command of the R4 interface. All responses of the R4 interface contain a pair of statuses (AIMStatus and AuthStatus) with the following semantics:

- AIMStatus is the bearer of the authentication session status (see Appendix A).
- AuthStatus is the bearer of the authentication handshake result (see Appendix B).

The integrator should always take this pair into consideration when checking success.

The following figure shows the status diagram of the authentication session along with the R4 interface operations, which influence the actual process:

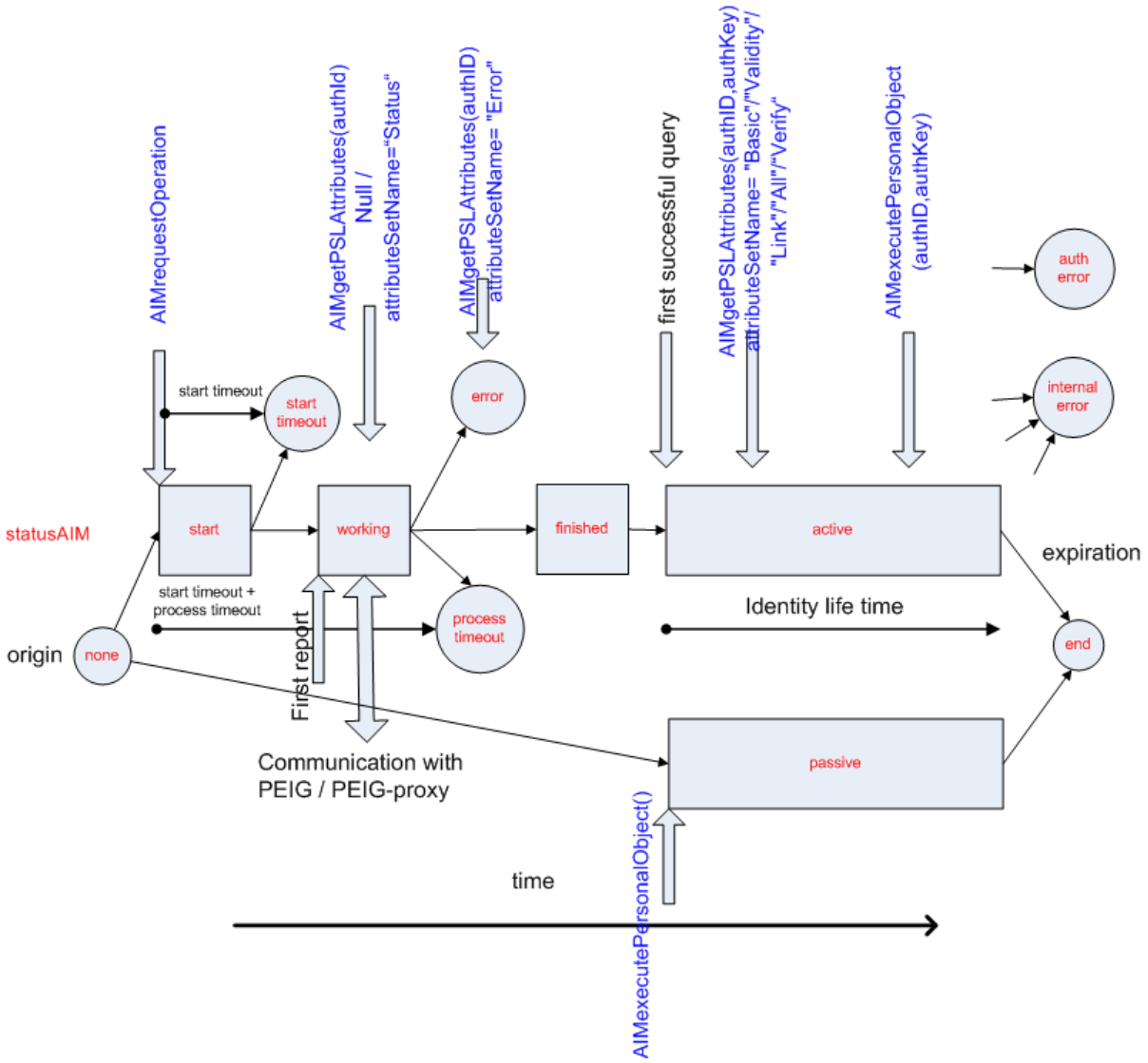

Figure 4-1: Status diagram of the authentication session

5. Integration and changes in existing applications

This chapter describes two methods of integration with existing applications. Since most applications require linking the electronic identity to a user profile, this structure must persist somewhere. ADUCID thus supports two integration scenarios:

1/ The link is stored in the ADUCID database
2/ The link is stored in the integrating application's database

Individual integration approaches are covered in chapters below.

5.1. The link is stored in the ADUCID database

Each application can own a set of attributes (user attribute set) in ADUCID, and store any information in them (so-called directory personal objects). In this integration scenario, the primary key of the integrating application user is stored in the set of attributes. After storing this information, the AIMexecutePersonalObject command is used with the R4 interface's write method.

After successful authentication, the application can read the set of attributes by calling the AIMexecutePersonalObject command with the read method, and pair the primary key with the user in the user's database.

To create a set of attributes for the given application and for enforcing uniqueness of the primary key, it is necessary to contact the ADUCID administrator (see “ADUCIDServerKit Administration Guide”).

The integration scenario is schematically illustrated in the following figure.

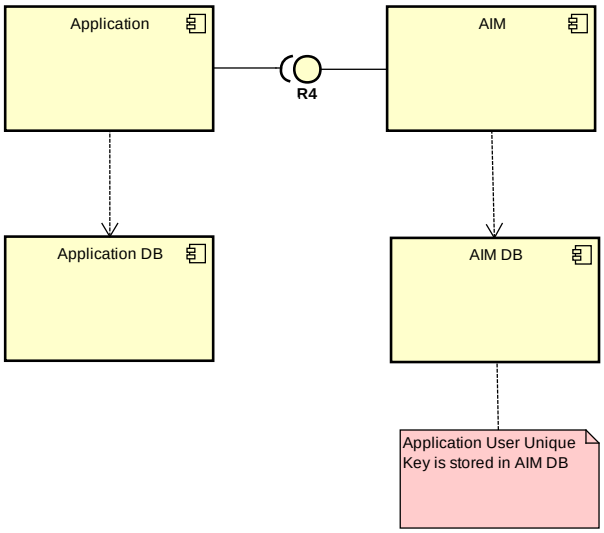

Figure 5-1: Integration without modification of the application database

5.2. The link is stored in the integrating application’s database

Each identity has a unique identifier within the ADUCID framework (so-called User directory index - UDI). In this scenario, the application has to save this identifier at the user. The UDI attribute is available after successful authentication by calling the AIMgetPSLAttributes command of the R4 interface.

The integration scenario is schematically illustrated in the following figure.

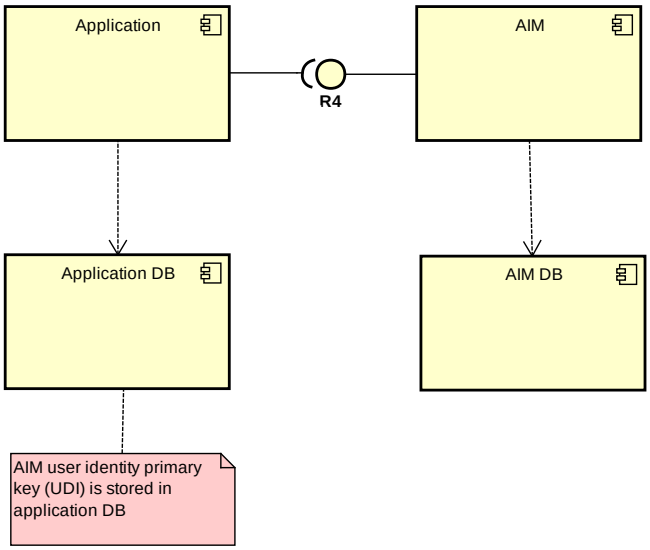

Figure 5-2: Integration without modification of the AIM database

6. Appendix A

The following table lists the semantics of individual statuses of the authentication session (AIMStatus):

Status	Brief description	Note
none	Initial status	Initial status of the status diagram.
start	Status that occurs when the authentication session is generated on AIM	From this point, it is possible to inquire about the status of the authentication process using authId.
startTimeout	Final status - PEIG did not start communicating within the defined time frame	This status typically means that there is a problem either in communication between PEIG/AIM, or that PEIG-Proxy is not available.
working	Authentication handshake between PEIG and AIM was successfully initiated.	This status means that the authentication handshake has begun.
processTimeout	Final status - the authentication handshake was not performed within the defined time frime	This status typically means that the responses from PEIG are slow or the user's reaction when approving a request is slow.
error	Final status - problem during authentication	"Masked" problem during authentication. The specific reason can be obtained by a subsequent query via the AIMGetPSLAttributes command of the R4 interface with attributeSet="Error". A list of possible return statuses is provided in the Appendix B section.
finished	Authentication handshake was successful	The authentication process was performed without problems. The operation result is available on the server. The operation result can be obtained using a pair (authId, authKey).
active	Authentication session was verified	Verification was performed and the authentication session is prepared to work for a period of time defined by the system.
passive	Passive status	Used on the secondary AIM during the identity link operation.
end	Authentication session ended	The authentication session has been terminated by the program, or the time period defined for work with an open session has expired.
internal-error	Undefined error	Unspecified error.
Auth-error	Error in authentication secret	The authentication handshake ended and the valid authentication secret authKey is not used.
Client-binding	Binding is executed	The communication before the start of an operation enforcing the binding mode and generating the needed information
KeepAlive	Wait pseudo-status.	A refresh of the communication channel is needed.

Table 2: Semantics of individual statuses of the authentication session (AIMStatus)

7. Appendix B

This appendix provides a complete list of potential results of operations within the authentication process framework.

Code	Brief description	Note
OK	Success	Normal -> normal success

Code	Brief description	Note
KO	Failure	Normal -> normal negative result
NAU	Disagreement at user side	Rejected by user - real or artificial because PEIG is temporarily blocked -> normal authentication failure
PPNP	PEIG is not available	Functional problem - PEIG is not connected to PEIG-Proxy -> Display to user (with instructions)
USP	Unknown service provider	Normal status - unknown user -> management (initialization menu - init)
USSP	Unknown secondary service provider	Normal status - unknown user for secondary provider -> management (explanatory text - link with other SP)
UU	Unknown user	Operating status or security problem - the identity obtained from PEIG does not exist in AIM -> management - either it is an attack, or repair after AIM failure - reinit
UUS	Unknown user for secondary provider	Operation status or security problem - the identity does not exist in a secondary AIM -> management - it is either the operation status, attack or repair after failure of AIM - init, or reinit at secondary AIM
UI	Invalid identification	Operation status - PEIG Operation status - PEIG or AIM determined that CyberID is violating terms of validity (time or number of uses) -> management - e.g. rechange menu or change in access rights
UPR	Unsupported security profile	AIM or PEIG cannot find (accept) the requested security profile -> change of profile or denial of PEIG
UIP	Unsupported IL security profile	AIM or PEIG cannot find (accept) the requested IL (identity link) security profile -> change in IL profile or request denial
UOP	Unsupported extension object profile	PEIG cannot accept the required security profile of the extension object -> change of profile or request denial
VI	Valid identity	Functional problem or security incident - CyberID is valid upon requirement for rechange -> probable use of rechange
NEO	Non-existing extension object	Functional problem - request for non-existing object -> probable application error
NTD	Nothing to do	Normal - nothing to apply the request to -> normal behaviour - failure (e.g. reading an empty list of objects) - application error or normal operation status
SPE	Error of second PEIG	Security of functional problem - attack or operation status or user error
UTL	Unsupported transition between security levels	Cannot accept the required change in security profile -> change of request or acceptance of unknown CyberID
DLN	Requested login name is already used by another PEIG	Functional problem - duplicity of legacy login name -> probable error of application or application user
NER	Insufficient rights to process the request	Functional problem -> application error or user problem
DR	Duplicate replica	Functional problem -> application error or user problem
NS	Non-existing session	Functional problem -> application error or user problem
CTO	Exceeded max. communication time	Functional problem -> configuration error or operation problem
ERR	Unspecified error	Security of functional problem: attack or implementation error, standard, general error message instead of specific ones

Code	Brief description	Note
UV	Unsupported version	Functional problem (incompatibility) -> management - change of request
DI	Repeated initialization	Functional problem or security incident - PEIG identified an attempt at repeated origin of a CyberID for the same AIM -> either an error in the application that did not check the existence of the CyberID, or a security attack or configuration error - another AIM exists with the same SPID
CR	Applying a rejected change	Recovery from problem - consequence
MI	Missing identity	Functional problem or security incident - CyberID does not exist upon request for reinit -> probable incorrect use of reinit
IE	Self-termination	PEIG security compromised-> management of the compromise, e.g. flagging an attribute and blocking access rights
NAP	Not accepted by PEIG	Secondary error - use primary
UCC	Incompatible keys	Functional problem - problem with keys in extension object -> probable application error
NOP	No operation requested	Functional problem or attack -> probable application error
UIL	Unknown ILID	Functional problem or attack -> probable application error
ILM	Missing ILID	Functional problem or attack -> probable application error
ISE	Identity Link electronic stamp error	Functional problem or attack -> probable application error
NSA	Missing address of secondary AIM	Functional problem or attack -> probable application error
NU	Not unique	Internal alarm or functional or security problem
LI	Locked identity	Normal - identity removed from use by administrator or automatically -> normal authentication failure
DMR	Duplicated meeting room	Normal—meeting room name conflict, the attempt to create a second meeting room with the same name results in a normal failure
UMR	Unknown meeting room	Normal—meeting room name conflict, the attempt to enter the non-existent meeting room results in a normal failure
CMR	Closed meeting room	Normal—the attempt to enter into a closed meeting room results in a normal failure
MET	Meeting room enter timeout	Normal—the second PEIG did not enter into the meeting room in time results in a normal failure
MCT	Meeting room confirmation timeout	Normal—the first PEIG did not confirm the second PEIG in time results in a normal failure
BIM	Binding item is missing	Security or functional problem—the required binding information is missing
BEE	Binding evaluation error	Security or functional problem—an attack to a banded channel or target application integration error
UBM	Unable binding mode	Security or functional problem—probably a target application integration error
MAD	MITM attacker detected	Security problem—the MITM attacker is recognised
BTO	Binding timeout	Normal—timeout in binding communication

Code	Brief description	Note
PCD	PEIG copy detected	Security problem—the PEIG copy is recognised
ADM	Anti-copy data missing	Security or functional problem—probably an attack or internal error or HW error
AAF	AIM anti-copy failure	Normal—AIM tolerated storage failure has been recognised by anti-copy detection
PAF	PEIG anti-copy failure	Normal—PEIG tolerated storage failure has been recognised by anti-copy detection
ACI	Anti-copy check impossible	Security or functional problem—probably an attack or internal error
LLF	Locked local factor	Normal or Security problem— Local Factor is locked by time lock
ALF	Absent local factor	Normal – requested Local factor does not exist -> normal behaviour - failure - application error or normal operation status
ULT	Unsupported local factor technology	Normal – the PEIG HW does not support requested Local factor -> normal behaviour, user issue
DLI	Duplicated local factor init	Normal – repeated initialization of the Local factor -> normal behaviour - failure - application error or normal operation status
ULF	Unverified local factor	Internal status
SLS	Successful local factor synchronization	Internal status
ICF	Integrity check failed	Security or functional problem—probably an attack or internal error
NTE	Network error	Normal—communication infrastructure error
SCE	Secondary communication error	Normal—communication infrastructure error
EOP	Empty original PEIG	Normal – the source (primary) PEIG in is empty -> normal behaviour - failure - user or application error or normal operation status
URM	Unsupported replica mode	Normal – requested replica is not possible -> normal behaviour - failure - application error or normal operation status
BLR	Blocked Replica	Normal or security problem – requested replica was blocked -> normal behaviour - failure - user error or attack
LFM	Local Factor merge	Internal status
COR	Confirmed Replica	Internal status
WFC	Waiting for confirmation	Internal status
DLV	Do Local Factor verification	Internal status
MRR	Meeting Room Ready	Internal status
WBK	Waiting for Binding Key	Internal status
HCM	HW Check item missing in auth vector	Security or functional problem—probably an attack or internal error or HW error
HCD	PEIG HW change detected	Security problem—the PEIG hardware change is recognised
SUV	Secondary (system) Unsupported Version	Normal – Secondary AIM version is old and it is not supported.
OPV	Old Peig Version	Normal – the PEIG version is old and do not support requested functionality
USF	Unsupported Feature	Normal – the requested features set is not supported

Code	Brief description	Note
INR	Insufficient rights	Administration/proofing – you have insufficient rights to access an operation
DNF	Data not found	Administration/proofing – requested data not found on AIM
DAE	Data already exists	Administration/proofing – requested data already exists on AIM
CNF	Code not found	Administration/proofing – requested activation code not found on AIM
FNF	Form not found	Administration/proofing – requested form identifier not found on AIM

Table 3: Error statuses of authentication process

8. Abbreviations

The following is a summary of used abbreviations and their meaning:

ADUCID®

ADUCID® (Automatic Liberal and User Centric Electronic Identity) is a new authentication system that functions on the principle of providing services and infrastructures of electronic identities. It is an identification and authentication framework based on new ideas, rules, procedures and implementations for work and support of a unified method of authentication.

The main purpose of ADUCID® is to provide identification and authentication services in the cybernetic world of ICT systems using the ADUCID® secure authentication layer.

ADUCID® provides:

- Electronic identity services
- Secure authentication services
- Essential infrastructure for listed services

PEIG®

PEIG® (Personal Electronic Identity Gadget) is a device that provides full management capabilities for its user's electronic identities. Using the user identity, it also provides automatic authentication between the client application (used by the user) and the server part of the target application (that the user is accessing).

AIM

ADUCID® Identity Machine - Implements ADUCID® server functionality itself. It performs all ADUCID® operations and provides access to user data stored along with electronic identities in the LDAP database.

Through a standard network interface (web services), it provides target applications with services related to administration of electronic identities.

AIM-Proxy

Specialized module for web applications used to communicate with the client web browser upon authentication of HTML applications. This component enables ADUCID® login into your application without modifying the browser (redirect login).

PEIG-Proxy

Specialized software communications module that connects PEIG Core to the client target application and AIM. It also functions as an application firewall to protect PEIG® Core. It must be run on the same computer as the client part of the target application.

9. Literature

[1] *ADUCID Architecture*