Legal Notice
All information included in this document is subject to change without notice. The Members of the TAS3 Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the TAS3 Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.
## The TAS³ Consortium

<table>
<thead>
<tr>
<th>Beneficiary Name</th>
<th>Country</th>
<th>Short</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>K.U. Leuven</td>
<td>BE</td>
<td>KUL</td>
<td>Project Mgr</td>
</tr>
<tr>
<td>Synergetics nv/sa</td>
<td>BE</td>
<td>SYN</td>
<td>Partner</td>
</tr>
<tr>
<td>University of Kent</td>
<td>UK</td>
<td>KENT</td>
<td>Partner</td>
</tr>
<tr>
<td>University of Karlsruhe</td>
<td>DE</td>
<td>KARL</td>
<td>Partner</td>
</tr>
<tr>
<td>Technische Universiteit Eindhoven</td>
<td>NL</td>
<td>TUE</td>
<td>Partner</td>
</tr>
<tr>
<td>CNR/ISTI</td>
<td>IT</td>
<td>CNR</td>
<td>Partner</td>
</tr>
<tr>
<td>University of Koblenz-Landau</td>
<td>DE</td>
<td>UNIKOLD</td>
<td>Partner</td>
</tr>
<tr>
<td>Vrije Universiteit Brussel</td>
<td>BE</td>
<td>VUB</td>
<td>Partner</td>
</tr>
<tr>
<td>University of Zaragoza</td>
<td>ES</td>
<td>UNIZAR</td>
<td>Partner</td>
</tr>
<tr>
<td>University of Nottingham</td>
<td>UK</td>
<td>NOT</td>
<td>Partner</td>
</tr>
<tr>
<td>SAP Research</td>
<td>DE</td>
<td>SAP</td>
<td>Coordinator</td>
</tr>
<tr>
<td>Eifel</td>
<td>FR</td>
<td>EIF</td>
<td>Partner</td>
</tr>
<tr>
<td>Intalio</td>
<td>UK</td>
<td>INT</td>
<td>Partner</td>
</tr>
<tr>
<td>Risaris</td>
<td>IR</td>
<td>RIS</td>
<td>Partner</td>
</tr>
<tr>
<td>Kenteq</td>
<td>BE</td>
<td>KETQ</td>
<td>Partner</td>
</tr>
<tr>
<td>Oracle</td>
<td>UK</td>
<td>ORACLE</td>
<td>Partner</td>
</tr>
<tr>
<td>Custodix</td>
<td>BE</td>
<td>CUS</td>
<td>Partner</td>
</tr>
<tr>
<td>Symlabs</td>
<td>PT</td>
<td>SYM</td>
<td>Partner</td>
</tr>
</tbody>
</table>

**Disclaimer:** This document has not been reviewed or approved by European Commission.

## Contributors
<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph Alhadeff</td>
<td>Oracle</td>
</tr>
<tr>
<td>Brendan Van Alsenoy</td>
<td>KUL</td>
</tr>
<tr>
<td>Guglielmo De Angelis</td>
<td>CNR/ISTI</td>
</tr>
<tr>
<td>Michele Bezzi</td>
<td>SAP</td>
</tr>
<tr>
<td>David Chadwick</td>
<td>KENT</td>
</tr>
<tr>
<td>Brecht Claerhout</td>
<td>CUS</td>
</tr>
<tr>
<td>Danny De Cock</td>
<td>KUL</td>
</tr>
<tr>
<td>Marc Van Coillie</td>
<td>EIF</td>
</tr>
<tr>
<td>Elisa Costante</td>
<td>TUE</td>
</tr>
<tr>
<td>Ioana-Georgiana Ciuciu</td>
<td>VUB</td>
</tr>
<tr>
<td>Carlos Flavian</td>
<td>ZAR</td>
</tr>
<tr>
<td>Michal Guinaliù</td>
<td>ZAR</td>
</tr>
<tr>
<td>Seda Güurses</td>
<td>KUL</td>
</tr>
<tr>
<td>Ingo Dahn</td>
<td>UNIKOLD</td>
</tr>
<tr>
<td>Thorsten Haberecht</td>
<td>KARL</td>
</tr>
<tr>
<td>Jerry den Hartog</td>
<td>TUE</td>
</tr>
<tr>
<td>Jeroen Hoppenbrouwers</td>
<td>SYN</td>
</tr>
<tr>
<td>Sampo Kellomäki (main contributor)</td>
<td>Unaffiliated</td>
</tr>
<tr>
<td>Thomas Kirkham</td>
<td>NOT</td>
</tr>
<tr>
<td>Michael Kutscher</td>
<td>UNIKOLD</td>
</tr>
<tr>
<td>Keqin Li</td>
<td>SAP</td>
</tr>
<tr>
<td>Stijn Lievens</td>
<td>KENT</td>
</tr>
<tr>
<td>Gilles Montagnon</td>
<td>SAP</td>
</tr>
<tr>
<td>Jutta Mülle</td>
<td>KARL</td>
</tr>
<tr>
<td>Jens Müller</td>
<td>KARL</td>
</tr>
<tr>
<td>Jean-Christophe Pazzaglia</td>
<td>SAP</td>
</tr>
<tr>
<td>Lex Pohlman</td>
<td>KENTEQ</td>
</tr>
<tr>
<td>Quentin Reul</td>
<td>VUB</td>
</tr>
<tr>
<td>Brian Reynolds</td>
<td>RIS</td>
</tr>
<tr>
<td>Marc Santos</td>
<td>UNIKOLD</td>
</tr>
<tr>
<td>Luis Schilders</td>
<td>CUS</td>
</tr>
<tr>
<td>Magali Seguran</td>
<td>SAP</td>
</tr>
<tr>
<td>Silvia von Stackelberg</td>
<td>KARL</td>
</tr>
<tr>
<td>Slim Trabelsi</td>
<td>SAP</td>
</tr>
<tr>
<td>Luk Vervenne</td>
<td>SYN</td>
</tr>
<tr>
<td>Sara Winfield</td>
<td>NOT</td>
</tr>
<tr>
<td>Gang Zhao</td>
<td>VUB</td>
</tr>
</tbody>
</table>
Contents

LIST OF FIGURES ........................................................................................................... 9

EXECUTIVE SUMMARY ............................................................................................... 11

1 INTRODUCTION .......................................................................................................... 13
  1.1 TAS³ ARCHITECTURE AT GLANCE ................................................................. 13
  1.2 METHODOLOGY ................................................................................................. 14
  1.3 NORMATIVE CLAIM ......................................................................................... 15
  1.4 REVIEW OF PREVIOUS WORK ...................................................................... 15
  1.5 READER’S GUIDE ............................................................................................ 16

2 TAS³ HIGH LEVEL ARCHITECTURE ....................................................................... 17
  2.1 OVERVIEW ........................................................................................................ 17
  2.2 BASIC ARCHITECTURAL ENTITIES .............................................................. 18
     2.2.1 Major Components ................................................................................... 18
     2.2.2 Enforcement Points on Web Service Call Path ........................................ 21
     2.2.3 Authorization Subcontinent ...................................................................... 22
  2.3 MAJOR FLOWS: FRONT CHANNEL AND BACK CHANNEL ........................... 24
  2.4 OVERVIEW OF DATA MODELS ..................................................................... 25
     2.4.1 Federation Relations for Core Security Architecture ......................... 25
     2.4.2 Personal Data and Applications ............................................................... 27
     2.4.3 Using Sticky Policies to Protect Data ...................................................... 27
     2.4.4 Using Encryption to Protect Data ............................................................ 27
  2.5 AUTHORIZATION PROCESS ........................................................................... 27
  2.6 ENFORCEMENT PROCESS .............................................................................. 28
  2.7 CONFIGURATION PROCESS .......................................................................... 30
  2.8 AUDIT ................................................................................................................. 31

3 CORE SECURITY ARCHITECTURE ........................................................................... 35
  3.1 FLOWS .............................................................................................................. 35
  3.2 TOKENS, ACCESS CREDENTIALS ................................................................. 36
     3.2.1 Attribute Pull Model ............................................................................... 36
     3.2.2 Linking Service: Attribute Push Model .................................................. 44
     3.2.3 Simple Attribute Push Model .................................................................. 48
  3.3 DELEGATION ...................................................................................................... 49
     3.3.1 Invitation Based Token Approach ............................................................ 51
     3.3.2 Mandate Tokens Approach ..................................................................... 55
     3.3.3 Delegation by Direct Authorization Rule ................................................. 56
     3.3.4 Delegation by Role Based Authorization Rule ........................................ 56
     3.3.5 Token Based Delegation to Well Known Delegatee ............................... 57
3.3.6 Token Based Delegation of a Received Role .............................................. 57
3.3.7 Multi-layer (Chained) Delegation .......................................................... 58
3.4 SUBJECT OF THE PII NOT PRESENT - TRANSACTION .......................... 58
3.5 BREAK-THE-GLASS AUTHORIZATION .................................................... 59
3.6 TRUST AND PRIVACY NEGOTIATION ...................................................... 59
3.7 INTEROPERATION ACROSS TRUST NETWORKS ..................................... 60
   3.7.1 Semantic Interoperability Engine ..................................................... 60
3.8 PROPERTIES OF WEB SERVICE BINDING ............................................. 61

4 APPLICATION SPECIFIC ARCHITECTURE ................................................. 62
   4.1 PROTOCOL SUPPORT FOR CONVEYANCE OF STICKY POLICIES .............. 62
   4.2 LEGACY INTEGRATION STRATEGY ....................................................... 63
   4.3 ADPEP ............................................................................................. 64
   4.4 REPUTATION FEEDBACK .................................................................. 65
   4.5 BUSINESS PROCESS REGISTRATION ............................................... 66

5 USING BUSINESS PROCESS MODELLING TO CONFIGURE THE COMPONENTS ... 67

6 OVERSIGHT AND MONITORING ................................................................. 69
   6.1 DASHBOARD .................................................................................... 69
   6.2 RIGHT OF ACCESS, RECTIFICATION, AND DELETION ........................... 69
       6.2.1 Identification of Originating Authority ....................................... 70
       6.2.2 Facilitating Self Service Interface to Right of Access .................... 70
       6.2.3 Propagation of Rectifications by the Originating Authority ............ 70
   6.3 ON-LINE COMPLIANCE TESTING ....................................................... 71
       6.3.1 Involved Actors ...................................................................... 72
       6.3.2 On-line Testing Process and Architecture ................................... 72
   6.4 OPERATIONS MONITORING AND INTRUSION DETECTION ................... 76
   6.5 LOG AUDIT ..................................................................................... 76
       6.5.1 Log Collection and Storage ....................................................... 77
       6.5.2 Privacy Issues: What to Collect and What to Report .................... 78
   6.6 FORMAL COMPLIANCE AUDITS ....................................................... 78
   6.7 ADMINISTRATIVE OVERSIGHT ......................................................... 78

7 CONCLUSION: TAS³ IS SECURE AND TRUSTWORTHY ................................. 79

ANNEX A ENUMERATION OF AUDIT EVENTS ............................................. 80

ANNEX B TAS³ RISK ASSESSMENT ............................................................. 84
   B.1 EXECUTIVE SUMMARY ................................................................. 84
   B.2 INTRODUCTION ............................................................................. 84
   B.3 RISK ANALYSIS METHODS ........................................................... 84
       B.3.1 OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) .......... 84
B.3.2 CRAMM ................................................................. 85
B.3.3 Microsoft’s Security Risk Management ........................................... 85
B.3.4 CORAS ................................................................. 85
B.3.5 ISO/IEC 27001 (BS7799-2:2002) ................................................. 86
B.4 OUR METHODOLOGY .......................................................... 86
B.5 RISK ASSESSMENT METHODOLOGY .............................................. 87
B.5.1 Identify the Components ....................................................... 87
B.5.2 Identify the Sub Component .................................................... 89
B.5.3 Identify the assets ............................................................ 89
B.5.4 Identify the Threats ........................................................... 89
B.5.5 Identify the Attacks .......................................................... 91
B.5.6 Identify the mitigation ......................................................... 91
B.6 T3-ACVS: AUTHORIZATION CREDENTIAL VALIDATION SERVICE .......... 94
B.7 T3-BP-CLIENT: BUSINESS PROCESS USER INTERFACE ....................... 94
B.7.1 Entry points ............................................................... 94
B.7.2 Asset: State of process instances, confidentiality and integrity of process data ........ 95
B.8 T3-BP-DEL: BUSINESS PROCESS DELEGATION SERVICE .................. 95
B.8.1 Entry points ............................................................... 95
B.8.2 Asset: Integrity of user-task assignment .................................... 95
B.9 T3-BP-ENGINE-ODE: APACHE ODE BUSINESS PROCESS EXECUTION ENGINE .... 96
B.9.1 Entry points ............................................................... 96
B.9.2 Asset: Executable process models ......................................... 96
B.9.3 Asset: Running process instances (state and data) ....................... 96
B.10 T3-BP-ENGINE-ODE: APACHE ODE BUSINESS PROCESS EXECUTION ENGINE .... 96
B.10.1 Entry points ............................................................. 96
B.10.2 Asset: Executable process models ....................................... 97
B.10.3 Asset: Running process instances (state and data) .................... 97
B.11 T3-BP-MGR: BUSINESS PROCESS ADMINISTRATION INTERFACE ........ 97
B.11.1 Entry points ............................................................. 97
B.11.2 Asset: Status of business process instances ............................... 97
B.12 T3-BP-PEP: BUSINESS PROCESS POLICY ENFORCEMENT POINT ............. 98
B.12.1 Entry points ............................................................. 98
B.12.2 Asset: Correct policy enforcement ...................................... 98
B.13 T3-BP-PIP: BUSINESS PROCESS POLICY INFORMATION POINT ............. 98
B.13.1 Entry points ............................................................. 98
B.13.2 Asset: Correct information for policy enforcement ..................... 99
B.14 T3-BP-SMC: BUSINESS PROCESS SECURITY CONFIGURATION COMPONENT .... 99
B.14.1 Entry points ............................................................. 99
B.14.2 Asset: Business process security annotations and security ............. 99
B.14.3 Asset: Security enhanced business process models and business process security configurations generated by the SMC component ......................... 100
B.15 T3-BUS-AUD: Audit Event Bus ................................................................. 100
    B.15.1 Entry points ..................................................................................... 100
    B.15.2 Asset: Channels of audit event type .............................................. 100
    B.15.3 Asset: Status of audit event type ................................................... 100
B.16 T3-DEL: Delegation Service ................................................................. 101
    B.16.1 Entry points ..................................................................................... 101
    B.16.2 Assets ............................................................................................... 101
    B.16.3 Threats .............................................................................................. 101
B.17 T3-IDP-ACSI S ......................................................................................... 102
    B.17.1 Entry points ..................................................................................... 102
    B.17.2 Data assets ....................................................................................... 103
    B.17.3 Nonfunctional assets ....................................................................... 103
    B.17.4 Attacks and mitigation ..................................................................... 103
B.18 T3-IDP-SHIB: Shibboleth IDP ............................................................... 103
    B.18.1 Entry Points ...................................................................................... 103
    B.18.2 Asset: Issued Credentials. ................................................................. 104
    B.18.3 Asset: Private Signing Keys of the Service ..................................... 104
    B.18.4 Asset: User Database (Attributes and Authentication Credentials) 104
    B.18.5 Asset: Session Information ............................................................... 106
    B.18.6 Asset: Configuration Files of the Service ....................................... 106
    B.18.7 Non-functional Asset: User Consent and Control of Data Release 106
    B.18.8 Non-functional Asset: Organisation Control of Data Release ...... 106
B.19 T3-IDP-SHIB-AGG: Enhancement to Shibboleth IDP to Support Attribute
    Aggregation ................................................................................................. 106
    B.19.1 Entry points ..................................................................................... 106
    B.19.2 Asset: The Audit Trail .................................................................... 106
    B.19.3 Asset: SAWS Records a Secure Audit Trail ................................. 107
    B.19.4 Asset: Private Keys for Signing/Encrypting ................................... 107
B.20 T3-LOG-SAWS: Secure Auditing Web Service .................................... 108
    B.20.1 Entry points ..................................................................................... 108
    B.20.2 Asset: The Audit Trail .................................................................... 108
    B.20.3 Asset: SAWS Records a Secure Audit Trail ................................. 109
    B.20.4 Asset: Private Keys for Signing/Encrypting ................................... 109
B.21 T3-LOG-WRAP-SAWS: Wrapper Web Service around SAWS with Extensions ................................................................................................. 110
    B.21.1 Entry points ..................................................................................... 110
    B.21.2 Asset: Audit Trails .......................................................................... 110
B.22 T3-OCT: Online Compliance Testing .................................................. 110
    B.22.1 Entry points ..................................................................................... 110
    B.22.2 Non-Functional Assets ................................................................... 111
B.23 T3-OCT-PLANNER: Test Planner for the Online Compliance Testing . . . 111
B.24 T3-PDP-BP: BUSINESS PROCESS POLICY DECISION POINT ............................ 111
B.25 TAS3-ONT-SER: ONTOLOGY SERVER ............................................................. 112
  B.25.1 Entry Points .............................................................................................. 112
  B.25.2 Asset: Ontology Server ............................................................................. 112
B.26 T3-PDP-BTG: BREAK THE GLASS PDP .......................................................... 112
  B.26.1 Entry points .............................................................................................. 112
  B.26.2 Asset: The BTG-State of the System ......................................................... 113
B.27 T3-PDP-M: POLICY DECISION POINT/MASTER PDP ................................... 113
B.28 T3-PDP-P: PERMISS PDP .................................................................................. 113
  B.28.1 Entry Points .............................................................................................. 113
  B.28.2 Asset: Authorisation Policy ..................................................................... 113
B.29 T3-PDP-T: TRUST POLICY DECISION POINT .............................................. 113
  B.29.1 Entry points .............................................................................................. 113
  B.29.2 Asset: Trust Policy ................................................................................... 114
  B.29.3 Asset: Trust-PDP Evaluation Service ....................................................... 114
B.30 T3-PEP-AI: APPLICATION-INDEPENDENT POLICY ENFORCEMENT POINT .... 115
  B.30.1 Entry Points .............................................................................................. 115
  B.30.2 Asset: Built-in Policies ............................................................................. 115
  B.30.3 Asset: Configuration File ......................................................................... 115
  B.30.4 Asset: Roots of Trust (PKCs) for Signature Verification .......................... 116
  B.30.5 Asset: System Coordinates Decision Making ......................................... 116
  B.30.6 Asset: The Sticky Store ............................................................................ 116
B.31 T3-POL-GUI: GRAPHICAL POLICY EDITOR ................................................. 116
  B.31.1 Entry points .............................................................................................. 116
  B.31.2 Asset: Information in Configuration File .................................................. 117
  B.31.3 Asset: Private Signing Key Used to Sign Policies ..................................... 117
B.32 T3-POL-CNL: CONTROLLED NATURAL LANGUAGE POLICY EDITOR ............ 117
B.33 T3-POL-WIZ: WIZARD POLICY EDITOR ....................................................... 117
B.34 T3-PORT-JBOSS: JBOSS PORTAL FRAMEWORK ........................................ 117
  B.34.1 Entry points .............................................................................................. 117
  B.34.2 Asset: user data (functional data) and user credentials (non-functional data) .... 118
  B.34.3 Asset: internal data (non-functional data) .............................................. 119
B.35 T3-REP-FEDORA: TAS3 REFERENCE REPOSITORY ...................................... 120
  B.35.1 Entry points .............................................................................................. 120
  B.35.2 Asset: user data (functional data) and credentials (non-functional data) .... 120
  B.35.3 Asset: internal data (non-functional data) .............................................. 121
B.36 T3-REP-JFEDORA: JFEDORA LIBRARY FOR TAS3 GENERIC DATA FORMAT .... 122
  B.36.1 Entry points .............................................................................................. 122
  B.36.2 Asset: Message based on Generic Data Format ....................................... 123
B.37 T3-SG-BASE: SOA GATEWAY BASE SYSTEM ............................................. 123
  B.37.1 Entry points .............................................................................................. 123
B.37.2 Asset: Backend Legacy data ................................. 123
B.37.3 Asset: Server Configuration Files ................................. 123
B.38 T3-SG-WSP: SOA GATEWAY WEB SERVICE PROVIDER ......................................................... 123
B.38.1 Entry points .......................................................... 123
B.38.2 Asset: Web services hosted by T3-SG-WSP component .................................................. 124
B.39 T3-SP-CVT: EUROPASS CV TRANSCODING WEB SERVICE ................................................... 124
B.39.1 Entry Points .......................................................... 124
B.39.2 Assets ................................................................. 124
B.39.3 Threats ................................................................. 124
B.40 T3-SP-MATCHER: JOB PROFILE MATCHING SERVICE ......................................................... 125
B.40.1 Entry points .......................................................... 125
B.40.2 Asset: Program logic that matches profiles of individuals to opportunities ................. 125
B.41 T3-STACK AND T3-SSO-ZXID .............................................. 125
B.41.1 Entry points .......................................................... 125
B.41.2 Data assets ........................................................... 125
B.41.3 Nonfunctional assets ............................................... 126
B.41.4 Attacks and mitigation ............................................... 126
B.42 T3-TPN-TB2: TRUST NEGOTIATION MODULE ................................................................. 126
B.42.1 Entry points .......................................................... 126
B.42.2 Assets ................................................................. 126
B.42.3 Threats ................................................................. 127
B.42.4 Attacks ................................................................. 127
B.43 T3-TRU-CTM: CREDENTIAL BASED TRUST SERVICE COMPONENT ........................................ 127
B.43.1 Entry points .......................................................... 127
B.43.2 Asset: Credentials .................................................. 127
B.43.3 Asset: CTM Web Service ........................................... 128
B.44 T3-TRU-KPI: KEY PERFORMANCE INDICATORS ............................................................ 128
B.44.1 Entry points .......................................................... 128
B.44.2 Asset: KPI Engine .................................................... 128
B.45 T3-TRU-RTM: REPUTATION TRUST MANAGEMENT SERVICE ............................................. 129
B.45.1 Entry points .......................................................... 129
B.45.2 Asset: Feedback ...................................................... 129
B.45.3 Asset: RTM Web Service ........................................... 129

ANNEX C USER CENTRICITY OF TAS3 ................................................. 131
C.1 EXECUTIVE SUMMARY ..................................................... 131
C.2 DEFINING ELEMENTS OF USER-CENTRICITY IN TAS3 ....................................................... 131
C.2.1 The user’s ability to express privacy preferences ......................................................... 131
C.2.2 The user’s ability to manage his own partial identities ................................................. 132
C.2.3 The user’s ability to express trust preferences and provide feedback ......................... 132
C.2.4 The user’s ability to view his personal data ............................................................... 133
C.2.5 Enhanced transparency ................................................................. 133

BIBLIOGRAPHY .................................................................................. 134
List of Figures

Figure 2.1: Using TAS³ top level model to start modelling of organizations that participate in Trust Network. ................................................................. 17
Figure 2.2: Major components of Organization Domain.................................................. 19
Figure 2.3: Front End calls Web Service, passing through 4 enforcement points (callouts, per Fig-2.2 of D7.1). ................................................................. 21
Figure 2.4: Recursive Web Service calls. .................................................................. 22
Figure 2.5: Authorization. ...................................................................................... 23
Figure 2.6: Front Channel and Back Channel Flows (the numbering indicates typical sequence of events). .............................................................. 24
Figure 2.7: Audit Event Bus (the numbering indicates typical sequence of events, the e-numbers indicate audit events). .................................................. 26
Figure 2.8: Authorization Process. ........................................................................ 28
Figure 2.9: Using model to configure Authorization Process. .................................. 29
Figure 2.10: Arrangement of enforcement points in web service call flow (numbered callouts, per Fig-2.2 of D7.1). ................................................................. 29
Figure 2.11: Pushing configurations from model. .................................................... 31
Figure 2.12: Configuration from Model (the numbering indicates typical sequence of events) ............ 32
Figure 2.13: Auditing an authorization decision. ..................................................... 33
Figure 2.14: Monitoring operation of the network using the configured model. .............. 34

Figure 3.1: General detailed flow of a service request .............................................. 35
Figure 3.2: Flow at front channel ........................................................................... 37
Figure 3.3: Front channel flow when using Identity Selector. .................................. 39
Figure 3.4: Flow of front channel call that makes a call on back channel. .................. 40
Figure 3.5: Single Sign-On (2,3), Discovery (4), and call to WSP (5). The blue ball represents discovery bootstrap. ......................................................... 41
Figure 3.6: Discovery Registration Using Front Channel Interface. ......................... 42
Figure 3.7: Flow of recursive calls on back channel. ............................................. 43
Figure 3.8: Linking Service: Registration phase. .................................................... 45
Figure 3.9: Linking Service: Login with attribute push phase. .............................................. 46
Figure 3.10: The enhanced login screen for attribute aggregation. ........................................... 46
Figure 3.11: Alice Prepares ePortfolio. ......................................................................................... 51
Figure 3.12: Bob using Alice’s Web Services by way of invitation. ........................................... 52
Figure 3.13: Invitation phase ........................................................................................................ 53
Figure 3.14: Accept invitation phase ............................................................................................. 55
Figure 3.15: Interoperation across contexts. ................................................................................. 60
Figure 4.1: Application Integration: PEP implemented directly in application. ............................. 63
Figure 4.2: Application Integration: ADPEP implemented in application itself. ............................ 63
Figure 4.3: Application Integration using ADPEP and (A) SOA Gateway, (B) WP9 DB as frontend to SOA GW, (C) WP9 database. .......................................................... 64
Figure 6.1: Overview of On-line Compliance Testing ................................................................. 71
Figure 6.2: UML Component Diagram of the On-line Compliance Testing (OCT). ........................ 74

Keyword List

Architecture, Security, Trust, Privacy
Architecture Executive Summary

This document contains version 1 of the TAS³ system architecture (by system architecture we mean the conceptual design that defines the structure and behaviour of a TAS³ trust network). As the Description of Work states, the TAS³ project’s main objective is to provide a next generation trust & security architecture that is ready to (1) meet the requirements of complex and highly versatile business processes, (2) enable the dynamic user-centric management of policies and (3) ensure end-to-end secure transmission of personal information and user-controlled attributes between heterogeneous, context dependent and continuously changing systems. This architecture has been designed to fulfill the above objectives through a combination of:

• providing users with the ability to meaningfully give their consent to the use of their personal information

• ensuring a complete set of audit information is recorded by a TAS³ trust network and that users have the ability to directly or indirectly see the audit information that pertains to their personal information. Note that there will not be a single central audit log. If a person needs to drill down into the distributed audit trail, he will need to be authorised and obtain sufficient permissions to access the various local audit logs in order to correlate the events and see the "big picture".

• a legal framework and set of model contracts that will contractually bind all service providers into operating in a trustworthy manner e.g. so as to honour the choices of users concerning the handling of their personal information

• a set of trusted third parties that facilitate the sharing of trust related information such as public keys, authorization attributes, and reputation information

• strong cryptographic algorithms and privacy preserving protocols

• end to end security through application layer encryption and digital signing

• sticky policies that cryptographically bind data and policies together, along with a policy enforcement infrastructure that controls access to all resources

• quality assurance and testing technology and actors to test if on-line services actually behave in compliance with their specifications.

This architecture document describes the conceptual entities that are needed and the services they should provide in order to operate a TAS³ trust network. These trust and privacy enhancing services include: authorization services, secure business process management services, delegation services, privacy preserving discovery services, identity management services, secure repository services and trust and reputation services. All of these services are usually needed regardless of the applications that might run in a TAS³ trust network. However, small centralized trust networks may be able to dispense with one or more of these trust and privacy enhancing services, e.g. discovery or delegation services, depending upon their requirements.

This architecture contains many novel features such as: a trust infrastructure based on novel metrics, actor behaviour and structural components which can be correlated together, an authorisation infrastructure which supports multiple policy languages and conflict resolution, an obligation infrastructure which enforces privacy throughout the trust network, and a distributed audit system which can be cross correlated with the necessary permissions. These are described in more detail in the specific work package deliverables.

The TAS³ architecture is designed to be standards, protocol, data and application agnostic so that any protocol capable of implementing the flows and satisfying the service requirements can potentially be
used by any application. Annex A maps these services onto the latest state of the art application independent protocols as far as is currently possible. This is to ensure interworking between the prototypes that will be developed in this project. Further standardization effort will be needed in order to fully complete this mapping and this will be documented in a future version of this architecture (or in other TAS³ deliverables).

Annex B shows an example deployment architecture that maximizes a service’s availability and is resilient to both system and network failures including denial of service attacks.

Annex C states the compliance requirements for participants in a TAS³ trust network. Legal, policy and technical compliance requirements are covered.

Annex D provides a set of use cases which allows the reader to see how an end user might use the services of a TAS³ trust network.

Annex E contains the first version of a business model that could be used to successfully operate a TAS³ trust network.

Annex F summarizes the threats that the TAS³ architecture is designed to protect against.

Annex G lists the events that should be captured in the secure audit trails of a TAS³ trust network.

Annex H gives some example protocol messages based on the mapping provided in Annex A.

Annex I provides a glossary of terms.

Scope. The TAS³ project has a narrower scope than the architecture that is documented here. This is natural as the novel research contributions of TAS³ are being made only in some areas of the architecture. However the full architecture needs to be documented as this will be needed both to successfully test the research results and to provide a production service. We present a comprehensive architecture that addresses actual use cases end-to-end, rather than simply an architecture of the services that are within the scope of our research.
1 Introduction

1.1 TAS³ Architecture at Glance

The TAS³ architecture provides the high level design of an infrastructure intended to provide the next generation of trust & security eco-systems that can (1) meet the requirements of complex and highly versatile business processes, (2) enable the dynamic user-centric management of policies and (3) ensure end-to-end secure transmission of personal information and user-controlled attributes between heterogeneous, context dependent and continuously changing systems.

The trusted architecture is built on three foundations: technical, policy and legal.

The technical architecture, introduced and described at a high level in this document, presents the different services that are needed in order to operate a trust network (or eco-system). Other work package deliverables provide more detailed designs of some of these services.

The technical architecture proposes a number of Policy Decision Points (PDPs) that are services capable of evaluating policies of various kinds and returning policy decisions to their callers - the Policy Enforcement Points (PEPs). The correct enforcement of user’s policies engenders trust in a network. Many policies in a TAS³ trust network will be sticky policies, meaning that the policy and the data to which it pertains, are cryptographically bound together, thereby ensuring that the policy is always there to be correctly enforced. Various types of policy and PDP are envisaged, trust PDPs, privacy PDPs, authorisation PDPs, delegation PDPs etc. Details of these PDPs and the policies they support will be provided in more detail in other workpackage deliverables e.g. from WP4, WP5, and WP7.

The legal framework and set of model contracts will be further developed in WP6. They are being designed to contractually bind all the service providers into operating in a trustworthy manner, for example, so as to honour all the choices of users concerning the handling of their personal information. As many trust enabling factors as possible will be built into the technical infrastructure described in this deliverable, thereby automating the controls and free organisations from the worry and overhead of ensuring that they do the right thing. When it is not possible to engender trust through technical controls alone, then legal controls through our model contracts will be used as the controls of last resort.

This architecture document describes a service oriented trust network. All the conceptual entities that are needed to form a trust and privacy preserving secure network operate as service providers and service consumers, and they collaborate together to provide the security services to end users. These trust, privacy and security services are application independent and are designed to ensure that whatever application the user is using, the application and its data are as secure, trustworthy and privacy preserving as is possible, given the risk assessment and cost constraints of the trust network. (We accept that absolute security is both technically impossible and financially unaffordable.)

The trust and privacy enhancing services offered by TAS³ include:

- authorization services, whose purpose is to answer the question "is this subject authorised to access this resource in this way"
- authentication services, whose purpose is to identify a subject and validate that the communicating party is indeed the identified subject
- privacy preserving services whose purpose is to provide pseudonymous identities for users and minimise the cross linking of identities
- trust negotiation services whose purpose is to determine if the remote communicating party is trustworthy enough to start a dialogue
- secure business process management services whose purpose is to ensure that business processes operate securely, and can be dynamically modified securely
- delegation services, whose purpose is to delegate credentials from a delegator to a delegate
• discovery services, whose purpose is to inform clients where particular services can be found

• trusted registries, whose purpose is to keep a directory of all services in the trust network who are known to provide services conforming to the TAS³ specifications

• attribute authorities whose purpose is to assert that particular users have particular attributes

• identity management services, which are a combination of an authentication service and an attribute authority

• secure repository services, whose purpose is to store users’ personal data securely and give users complete control over who should access their data and how they should handle it once they are given access to it

• trust and reputation services, whose purpose is to answer the question "how trustworthy is this actor (service provider or end user)"

• secure audit services whose purpose is to keep a tamper resistant record of transactions within the trust network so that legally admissible evidence can be obtained in the case of a dispute.

• on-line compliance testing services whose purpose is to ensure that all the services in a trust network comply with their published specifications and policies.

All of these services are usually needed regardless of the applications that might run in a TAS³ trust network. However, small centralized trust networks may be able to dispense with one or more of these trust and privacy enhancing services, e.g. discovery or delegation services, depending upon their requirements.

The TAS³ architecture is designed to be standards and protocol agnostic so that any protocol capable of implementing the message flows and service requirements of the conceptual service providers can potentially be used by any application. However, in order to ensure interworking between the prototypes being developed in this project, we have had to choose a subset of current state of the art protocols. Annex A maps (some of) our services onto the latest state of the art application independent protocols as far as is currently possible. Further standardization effort will be needed in order to fully complete this mapping and this will be documented in a future version of this architecture (or in other TAS³ deliverables).

1.2 Methodology

In presenting the architecture, we follow FMC (Fundamental Modeling Concepts) [FMC03] methodology for presenting the high level static structure. For flow diagrams we use a mixture of UML [UML2] sequence diagrams and ad-hoc “white boards”. The richness of the latter allow us to better convey relevant control flow and dataflow aspects simultaneously.

For more detailed descriptions we use UML [UML2] modelling, with occasional ad-hoc diagrams to clarify aspects that are not easily communicated using formalisms.

While we usually define, inline, the terminology we use, the authoritative definitions are in [TAS3GLOS]. All architecture documents use this same Glossary and it will not be duplicated in the individual documents.

The stakeholders in context of TAS³ Architecture are

• Users accessing their own data

• Professionals working on data of others

• Service Providers, TTP Operators, and Trust Guarantor (jointly Deployers)

• Security Officers

• Implementers

• TAS³ Members
The TAS³ mandate is to build secure, trustworthy, and user-centric technology ([TAS3DOW] section B.0 "Summary"), thus we have adopted methodology where every composition and flow includes a User facet. Most of the flows are viewed from the User perspective and the business and regulatory aspects are filled in from this perspective. Given that gaining trust of the Users is fundamental to wide spread adoption, we have opted to emphasize security, transparency, privacy, and user control when trading off efficiency and simplicity.

This document has two goals: (1) Act as an authoritative and prescriptive definition of the TAS³ architecture, and (2) communicate the architecture to the stakeholders, especially Deployers and Implementers. The latter goal is much in line with "Architect as Communicator" in Fig-1 of [FMC03].

1.3 Normative Claim

This document describes the TAS³ Architecture version 1 in a normative and prescriptive way. Any implementation or deployment claiming "TAS³ compliance" MUST abide by this document as well as [TAS3PROTO], and [TAS3COMPLIANCE]. A deployment usually has to satisfy additional requirements of the Trust Guarantor’s Governance or Consortium Agreement and certification procedures, some of which concern the software implementation and others the organizational properties. Use of TAS³ Brand is governed by a separate TAS³ Brand Agreement.

This document uses the keywords (MUST, SHOULD, etc.) of [RFC2119]. All text is normative unless expressly identified as non-normative. Prose and specification have precedence over examples, which in absence of normative text, should be considered RECOMMENDATIONS. Examples as used in the documents are illustrative of the application of the relevant principles contained in the documents and are not statements of principles.

This architecture, and related documents are copyrighted works of TAS³ Consortium members, as identified and dated. All Rights Reserved. This architecture, and related documents, are versioned and subject to change without notice. No warranty or guarantee is given. This architecture, and related specifications can be implemented on Royalty Free terms by anyone. However, no warranty regarding IPR infringement is given. For further details, please see [TAS3CONSOAGMT].

1.4 Review of Previous Work

TAS³ extends the State of the Art, as established by Identity Web Services Framework [IDWSF08], [HafnerBreu09], the Nessi Reference Architecture [NexofRA09], and Access-eGov Platform Architecture [AeGArch07]. [IDWSF08] includes a high level view, derived from documented requirements, and a low level implementable profile of various specifications backed up by interoperability and certification programs that verify interoperability in real life. [NexofRA09] only provides high level view and does not address identity issues (they even use term "federation" inaccurately, liable to cause confusion with Identity Federations) or interoperable protocol profiles - the definition of NEXOF Compliant Platform (NCP) is too vague and there are no interoperability or certification programs - [NexofRA09] fails to recognize clear prior art in [IDWSF08]. TAS³ extends the State of the Art by combining the web service, or SOA, framework with comprehensive authorization and trust management system, modelling domain, compliance validation (i.e. interoperability), and legal framework - in a whole that is concretely implementable. TAS³ addresses Long lifetime, Different Owners, and heterogenous IT environment concerns listed in [NexofRA09], Section 3.3. NexofRA discovery does not address discovery indexed by identity, though it does address discoverability by developers, which may be important for adoption.

[AeGArch07] architecture does not specify any concrete and interoperable implementation profile and its security details are vague. Never-the-less, they mention (but do not normatively reference) SAML SSO (no version), and WS-Security (no specific version or profile). They do recognize need for registry and discovery function, but do not discuss the interesting parts. Overall it appeared that their main ambition is not in architecture. They overviewed existing art and picked SOA and applied it to their problem domain.
using existing concepts without details research in the architecture area.

They use WSMO (http://www.wsmo.org/) based WSMX (Web Services Execution Environment).

The Web Service Modeling Ontology (WSMO) aims at describing Web Services in a machine understandable format, and thus enabling the automatic discovery, selection and composition of Web Service. As a result, WSMO provides a semantic to allow multiple organisations to cooperate for the completion of a service. For example, the Accreditation of Prior Learning (APL) process [TAS3D91PilotUC] requires multiple organisations to be contacted to build the portfolio of a candidate. WSMO is divided in four core components; namely ontologies, web services, goals, and mediators. The ontology element provides a syntax to describe ontological entities (e.g. concept, relation, axiom), which can then be used to represent the semantic of a domain of discourse. In other words, the ontology provides a common conceptualisation of the domain used the other WSMO components. The web service element semantically defines every aspect relevant to web services, such as functionalities and interfaces. For example, the functionality of a web service is expressed in terms of its capabilities and of the pre- and post-conditions associated to them. The goal element specifies the users’ objectives to be fulfilled by the execution of one or more web services. Finally, the mediator element establishes interoperability between mismatched resources. For example, it resolves mismatches in heterogeneous ontologies by finding mappings between their respective ontological entities.

1.5 Reader’s Guide

This document conforms to the TAS³ project-wide glossary [TAS3GLOS].

If you are a nontechnical reader you may want to start from Annex D to get overall understanding of the user experience, then skim the main document and perhaps consulting Figs 2.1 and ?? may be useful. You should also consult [TAS3BIZ] which gives a good motivation for the work shown here. You may also find [TAS3WP] and web site www.tas3.eu helpful in understanding the overall TAS³ concept.

If you are a researcher, this document is the right place to start to see where your research may fit within the architecture.

If you are a software developer you will want to read this document, but you will also want to read carefully [TAS3PROTO], which details protocol versions and gives suggestions about available open source packages that implement these protocols.

If you are a deployer, you should skim this document, perhaps look at [TAS3PROTO], and then work through [TAS3COMPLIANCE] as you prepare for your TAS3 certification.

If you are a reviewer, you should read Section 2 and then any other sections or annexes that interest you.
2 TAS³ High Level Architecture

2.1 Overview

Basic security measures. Secure encryption, message digest, and digital signature algorithms are used throughout where applicable. All Users and System Entities are authenticated to an appropriate degree. For the latter this means PKI authentication, but for the former anything from passwords to hardware tokens is possible. The details of these algorithms are not repeated here, but are covered in [TAS3PROTO] and [TAS3COMPLIANCE].

The TAS³ Architecture is a reusable overarching design that can be instantiated any number of times. It specifies a Trust Network (TN) and the manner in which the players, including Users and Service Providers, interact in the Trust Network. The TN may be composed of several organizations, mainly Service Providers (SPs), each of which may constitute a subnetwork and may participate in several other Trust Networks. The architecture addresses interaction of the subnetworks with each other and the top level Trust Networks. We also foresee multiple Trust Networks coexisting and interacting to various degrees. An organization can simultaneously belong to multiple TNs as long as it can simultaneously satisfy the requirements of each network.

Figure 2.1: Using TAS³ top level model to start modelling of organizations that participate in Trust Network.

Each Trust Network works in the legal context defined by its Governance Agreement. This architecture specifies some functions that are strictly necessary for protocol flows to work, and other functions that are necessary to satisfy nonfunctional properties like "secure" and "trustworthy". To impose on the players that the latter functions are implemented as well, we rely on legal obligation that stems from the Governance Agreement, as well as certification and audit programs, operated by the Trust Guarantor, to check that the legal obligations are met initially and on continued basis.
**TAS³ Trust Network Domain.** Consider Fig-2.1 where a Trust Network (TN), has chosen to adopt the overall TAS³ approach (which this and other documents specify). This means that at the "Summit" there is a Trust Guarantor (TG) who imposes on the TN the rules and model of operation. TG usually employs a **Security Officer** to maintain and enforce the model. The individual organizations may also have Security Officers responsible for their internal modelling and auditing.

**Model.** The Trust Network Domain configuration will be expressed using business process models, ontologies, and other models. The models are refined by each organization in their Modelling and Configuration Management. There will be several ontologies: architectural roles (e.g. Service Requester, Services Provider, Identity Provider), security ontology, privacy and data protection ontology and trust ontology. Payload services may define application specific ontologies, but they are not in scope of the TAS³ architecture. Ontologies in TAS³ are further discussed in [TAS3D2UPONTO]. Some mandatory policies emanating from EU will be modelled by the TAS³ project and incorporated to every TAS³ Compliant Trust Network Model (Req. D1.2-6.15-MinPolicy).

**Audit and Oversight.** The Trust Guarantor in its oversight role will operate compliance validation and audit functions. Each organization is expected to operate similar functions locally as **Audit & Monitor**. The audit trail stays principally within the organization, with Trust Guarantor only seeing pointers. There are some networkwide reporting and auditing requirements that guarantee that other parties in the network, and especially users, have enough transparency to operation of each party. This helps to transparently understand that what has happened is legitimate, prevent fraud, and increase overall trust in the network - a key business goal of TAS³.

**Runtime and Enforcement** concerns delivering the useful payload services, with appropriate mechanisms to authenticate and identify Users and Systems, as well as authorize the operations. Most of technical realization of TAS³ happens in this area.

**Cross Domain and Cross Context.** TAS³ Architecture expressly enables operation of services across domains. This can mean several organizations in one Trust Network, or it could even mean interworking of several Trust Networks.

### 2.2 Basic Architectural Entities

In this section we drill down in the static component view of TAS³ architecture.

#### 2.2.1 Major Components

Our architecture, see Fig-?? starts with User interacting with the Runtime & Enforcement area. Since TAS³ architecture is **user centric**, all action starts directly or indirectly with the User. Even offline, user-not-present, processes are seen to have been authorized by the User at some earlier time.

In the **Runtime** area, the User will interact with Payload services to obtain the tangible business benefits that motivated him to use the services in the first place. However, for the Payload to work in secure and trustworthy manner, services from Infrastructure and Discovery areas are needed. For the system as a whole to remain secure and trusted, functions in the Audit and Monitor area are needed. They will receive their input through Audit Event Bus of the Runtime environment.

**Front End Service.** User’s principal point of interaction with the system is a GUI, most commonly a Web GUI. This is a special kind of **Service Provider** that instead of speaking Web Services, e.g. SOAP, offers a user friendly interface. The Front End Services often call Web Services to perform all or parts of the functionality they provide. It is possible that the GUI is generated to match a Business Process Model.

**Web Service.** Machine accessible endpoint from which data or action services can be obtained. Machine-to-machine nature of Web Services is in contrast with the user-to-machine nature of the Front End Services.

The exact sequence of Web Services called will depend on a business process, whether expressly modelled or implicit to the design of the web services. A business process can encompass several Front Ends and the Web Services they call.

**Business Process Engine** is an orchestrating entity that controls how Front Ends and Service Providers, often Web Services, work together to achieve the objectives of the business process. It is depicted here as
Figure 2.2: Major components of Organization Domain.

being a separate service, but "in process" realizations are equally likely. In such case the Business Process
Engine would be inside the Front End Service, perhaps as linked in library. The role of the Business
Process Engine is to serve payload business processes. There is a similar Trust Network Process Manager
entity that, while technically similar, will exclusively execute business processes critical to the TN itself.

Dashboard is an important auditing and trust building feature of the TAS\textsuperscript{3} Architecture. It is a user
interface, a Web GUI, that allows the User to understand and audit how the system as a whole uses his
Personally Identifiable Information (PII). The Dashboard may also integrate a user interaction facility,
PII Consent Service, for asking users consent or other input that is required for a business process to
advance. A listing of the business processes in which the user has participated, or is participating cur-
rently is provided as well as a listing of Service Providers that hold data about the user or have handled
his data. All these features provide transparency. (Reqs. D1.2-2.11-Transp, D1.2-3.3-Dash, D1.2-6.3-
WhatHowWhyWho, and D1.2-12.15-Valid)

Identity Provider is the point where Users actually authenticate to the system. After authentication,
the IdP issues a Single Sign-On (SSO) token so that the Front End Service can complete the login process.
IdP has also an important role in providing Id Mapper bootstrap token for the User.

Authorization. This box actually represents an entire subcontinent of functionality. Authorization is
pervasive in TAS3 architecture. This topic is treated in more detail in Section 2.2.3.

Delegation provides mechanisms for one User to allow another User to use FE or WS services on his
behalf. Delegation also includes mechanisms for introducing users to one another, such as invites. In
some cases User can be replaced in delegation by a juridical person. In delegation both the delegator and
delegatee may be authenticated indirectly. A situation similar to delegation arises when User instructs a
service to act on his behalf. In this case the delegatee is a system entity, usually a Service Provider, and
is authenticated directly. The act-on-behalf delegation is handled by the ID Mapper component. (Req.
Trust Reputation encompasses a number of components that deal with gathering reputation data, usually via Audit Event Bus, and computing trust scorings that are then used in Authorization and Trust and Privacy Negotiator components. The trust and reputation system is also used to detect certain classes of fraud (Req. D1.2-7.21-Safe). The architecture and design of this subsystem is further elaborated in WP5 deliverables.

Trust Network Process Manager. There are many maintenance processes that a trust network must realize in order to work dynamically and react to threats rapidly. These include intake process for users (Req. D1.2-6.1-IntakePers), intake and certification process for organizations (Req. D1.2-6.2-IntakeOrg), and user’s access to his own data and audit trail (Req. D1.2-6.8-UserAccess). The application specific business processes belong to Business Process Engine, above.

Id Mapper is used to translate User’s IM token (Id Mapper bootstrap token) to a token usable for Web Service that is about to be called. Such translation is necessary as the user is known by different pseudonym at different services. This is used to express act-on-behalf relationships where Service Provider (delegatee) wields a token provided by Id Mapper (or in some cases by IdP). (Req. D1.2-2.3-BMs)

Registry Server contains knowledge about which end point serves which type of service for any given User. Typically Registry is queried as a preparatory step of web service call proper, but it could be queried in advance. (Req. D1.2-2.3-BMs)

Linking Service (aka Identity Aggregation) provides a facility for a user to indicate how he wishes his attributes to be aggregated.

Obligations Service (not depicted) provides a way to process many commonly occurring obligations such as data retention limit. Obligation handlers register with the obligations service. The service uses this information to advertise its capabilities in satisfying obligations. This leads to trust and privacy negotiation.

Trust and Privacy Negotiator. This is the server side of the negotiation. Every Service Requester, such as Front End Service, must implement Trust and Privacy Negotiator Client Agent (not shown in the figure). The Client Agent can be implemented as a web service and the Service Requester merely performs a web service call to the agent, which then engages in trust and privacy negotiation protocol with the web service provider’s Trust and Privacy Negotiator. Trust and Privacy Negotiator functions in many ways similar to the registry, but instead of returning all end points, only some are returned based on trust scoring.

Modelling and Configuration Management is connected to the TN level modelling. It also contains local ontologies, such as trust and privacy ontologies, and local Models and Configurations. All of these may be edited using Modelling Tools. From Models and Ontologies, configuration items can be generated and pushed to the Runtime using Management Event Bus, as governed by the Trust Network Process Manager.

An essential element of this architecture are community-managed ontologies (Model in Fig-??), which allows for unambiguous, but flexible, meaning agreement at all times. We can envisage several roles for these ontologies. It first provides a machine-understandable documentation of the architecture as well as a formal vehicle to exchange explicit semantic agreements (i.e. commitments) between partners and, eventually, systems. Thus, these commitments will enable the enforcement of (organisational and/or legal) policies within the TAS3 architecture. For example in Role-Based Access Control (RBAC), the role of a subject need to be provided with some semantics (e.g. a list of attributes) to be able to enforce authorization based on the privileges assigned to that role.

Secondly, the ontologies will assure that relevant parts of the system commit to the same interpretation of possibly ambiguous elements to allow for meaning alignment, certification and early conflict discovery. These ontologies will enable improved understanding; common methods of expressing terms enabling people and organisations to better trust each other in these application environments. TAS3 will integrate these architecture elements into a fully embedded trust framework to automate business processes managing personal information, which will result in considerable societal benefits. The Semantic
Interoperability Engine (Fig-3.15) will facilitate the interoperability across different contexts (e.g. across different organizations). Ontologies are further discussed in [TAS3D2UPONTO].

### 2.2.2 Enforcement Points on Web Service Call Path

![Diagram of Web Service Call Path]

**Figure 2.3:** Front End calls Web Service, passing through 4 enforcement points (callouts, per Fig-2.2 of D7.1).

Considering Fig-2.3, a Front End (FE) is composed of a Web GUI, a Web Application (the payload of the front end), and a Service Requester module which is used to call Web Services. The counter part of the Service Requester is the Service Responder module of the Web Service.

**Service Requester** is a software module that encapsulates the mechanics of performing a Web Service call. An implementation of the Service Requester module will be provided as a deliverable of the TAS³ Project. However, it is possible to implement this independently as long as all requirements prescribed here are maintained.

**Service Responder** is a software module that encapsulates the mechanics of accepting a Web Service call and responding to it. An implementation of the Service Responder module will be provided as a deliverable of the TAS³ Project. However, it is possible to implement this independently as long as all requirements prescribed here are maintained.

**Traffic Lights**

**PEPOut-Rq.** Service Requester Outbound Policy Enforcement Point (PEP). This PEP is used to check whether data can be submitted to the Web Service, or whether the call can be made at all. The PEP will contact organization’s Master PDP to obtain a policy decision.

**PEPIn-Rs.** Service Responder Inbound PEP. This PEP is used to check whether data or call can be accepted by the Web Service. It also records what obligations and policies does the Service Requester pledge to honour. These will be checked later by PEPOut-Rs.
**PEPOut-Rs.** Service Responder Outbound PEP. This PEP is used to filter the data on responder side and to perform any responder obligations attached to the data. In particular, the pledges recorded by PEPIn-Rs are checked against obligations and sticky policies attached to the data and if found unsatisfiable either data is filtered out or operation aborted. If no data can be returned, an error response will still be returned.

**PEPIn-Rq.** Service Requester Inbound PEP. This PEP is used to extract and perform or record for later performance any obligations attached to the response.

### Recursive Call

As shown in Fig-2.4, it is possible to chain web services calls, such that the application layer of upstream server may invoke as client a down stream service. There is no difference whether the Service Requester module resides in right hand side of a Front End or a Web Service, turned into Web Services Client (WSC).

This pattern can be repeated in any tree topology to any depth of call - however in practical implementation the call depth MAY be limited to 7 to avoid infinite recursion.

![Recursive Web Service calls.](image)

#### 2.2.3 Authorization Subcontinent

Authorization is everywhere in TAS³ Architecture. It often gets rolled up in small, but very meaningful symbol in the architecture. This is why we call authorization a "subcontinent" unto itself. It is described more fully in [TAS3D71IdMANAz]. This section addresses Reqs. D1.2-2.19-AzCredi, D1.2-2.20-Az, D1.2-4.5-ComplyPolicy, D1.2-4.6-BrkGlass, D1.2-6.4-Min, D1.2-7.6-Az.

Fig-2.5 depicts some of the components involved in the authorization. By far the most common case is that some payload service, such as a Front End or Web Service, needs to get an authorization decision and initiates the subflow.

**Policy Enforcement Point (PEP).** This is a software module usually built into the payload service. There are four fundamental types of PEP, as shown in Fig-2.3: in and out variants on Service Requester and Service Responder sides.
**Figure 2.5: Authorization.**

**Master Policy Decision Point (Master PDP).** The PEP calls Master PDP to obtain the authorization decision. Typically each organization will run a Master PDP (though other arrangements are possible). All logic of the authorization decision is masked behind the Master PDP. Thus the exact implementation details of Policy Decision Point Stack are irrelevant for the PEPs. The MasterPDP handles coordination and routing of requests to the PDPs in the stack and aggregates the authorization decisions received from the PDP. In a way it can be viewed as a PDP proxy with some smarts in it.

The Master PDP is responsible for arranging Break-the-Glass Authorization, see Section 3.5 and [TAS3D71IdMAnAz].

**Trust Network PDP** processes the policies that are coordinated at the Trust Network level. It can be implemented as a central Trust Network-wide service, or it can be distributed so that there is an instance of a Trust Network PDP at each SP, but the policies are centrally coordinated and pushed to the instances, perhaps using the Trust Network Process Manager.

**Organization PDP** processes the policies that an organization maintains. These policies may be over and above the the Trust Network-wide policies. The distinction from Trust Network PDP is maintained because the authority for deciding the policies is different.

**User PDP** function may implement User specific policies, i.e. policies set by the User. This could also involve evaluation of Sticky Policies. In practise, the User PDP may be implemented inside the Master PDP process.

**Trust PDP** is an interface to the Trust and Reputation Management subsystem which allows the Master PDP to query whether a contemplated action is acceptable from Trust and Reputation perspective. Such query has the advantage that the Trust and Reputation system does not need to disclose to the Master PDP the exact parameters that lead to this decision. The deliverables of WP5 will elaborate on structure and design of Trust PDP and Trust and Reputation System at large.
Credential Validation Service (CVS) is a subsystem that helps PEP to establish the validity of the credentials and attributes it is about to pass to the Master PDP. Typically these are received from front channel interaction or from an earlier web service call. The validation involves checking that they are properly signed and that PKI trust to the signing authority exists. Some namespace and syntax checks may be performed as well. The CVS may call on other components of the architecture to perform its functions.

Policy Information Point (PIP) is used to fetch additional attributes that may be needed for policy evaluation. PIP may call, in a recursive manner, on other components of the architecture to perform its functions. Special care needs to be taken in preventing infinite recursion and to ensure that the policies in the recursive levels allow the information to be returned for purpose of policy evaluation. PIP may be called either from PEP or from Master PDP. Exact choice is a question of optimization. The set of attributes needed for policy evaluation is difficult to determine. This is a research problem we hope to solve.

2.3 Major Flows: Front Channel and Back Channel

Implementable Flows. The flows we present are designed to be implementable with existing state-of-the-art protocols and software stacks. In particular standards based approaches are used for authentication, delegation, token passing, identity mapping, service discovery, authorization, and web services calls. Despite this, the present high level architecture is designed to be standards agnostic so that any protocol capable of implementing the flows and satisfying the requirements can potentially be used. See [TAS3PROTO] for details.

Figure 2.6: Front Channel and Back Channel Flows (the numbering indicates typical sequence of events).
From Fig-2.6 we can identify certain important principles (the authorization process is depicted in summary form as box "Az" to reduce clutter, see Section 2.5 for full description):

1. There can be any number of organizations in the Trust Network and each of these organizations may run a number of web sites (labelled as FE - Front End in the figure), Web Services (WS), and infrastructure services (sometimes called Trusted Third Parties).

2. Some architectural roles, like Identity Provider (IdP) can usefully be operated by several organizations in a Trust Network. The important point is that all the components are part of the model of the Trust Network and subject to its oversight.

3. Users will use their "home" IdP (e.g. IdP provided by their employer or educational institution) for Single Sign-On (SSO), but this does not prevent them from using web sites (labelled as FE - Front End in the figure, this is often called "front channel usage" or "user present scenario") of the other organizations (Req. 3.1 from D1.4), subject to access control decisions, of course.

4. The usage of a web site often triggers Web Services calls on the back channel. Finding out exactly which servers to contact and what credentials to use is handled by User’s Discovery and ID Mapper services ("IDMap" in the Fig-2.6) (Req. 8.1 from D1.4). Usually the Discovery Service is rather tightly coupled to the IdP.

5. It is feasible and common that Web Services can be called across organizational boundaries. Discovery and trust negotiation within the model set by the Trust Network will enable this to be possible.

6. When auditable events happen, in addition to local logging, a summary of the data is sent to the Audit Event Bus. Subscribed to the audit summaries are: (i) User’s Dashboard service so that the User can always see what happened and is in control; and (ii) the organizational and Trust Network audit layers. See blue arrows in Fig-2.7.

7. Although all organizations can potentially have all components, the fact that cross organization web site usage and service calls are explicitly provided for, makes it possible for an organization to outsource some, or all, of these services. Or the other way around, some organization may specialize in only providing the infrastructure services. This approach is often desirable to manage conflicts of interest.

This is a very flexible architecture and allows the responsibility for provision of services and infrastructure to be sliced and diced in many ways, according to business needs rather than technical limitations.

### 2.4 Overview of Data Models

#### 2.4.1 Federation Relations for Core Security Architecture

N.B. On first reading it may be advisable to skip this section as understanding of flows shown in Fig-3.4 will be useful.

One of the fundamental principles of the Core Security Architecture is use of federations, which may support persistent or transient identifiers. When correctly used, these types of identifiers allow privacy to be preserved by not leaking any correlation handles. This section addresses Reqs. \textit{D1.2-2.14-Priv}, \textit{D1.2-7.8-NoColl}, and \textit{D1.2-7.16-Nym}.

In order to implement persistent and pseudonymous federations, the IdP and IM have to keep state. In general, the federation table for an IdP that supports persistent pseudonymous identifiers will hold mappings as follows:

- \texttt{User at IdP1 --> [ encrypted pseudonym of user at SPA,}
- \texttt{encrypted pseudonym of user at SPB,}
- \texttt{...}
- \texttt{encrypted pseudonym of user at SPN ]}
The federation table for IM needs similar mappings

User’s pseudonym at IM --> [ encrypted pseudonym of user at SPA,
    encrypted pseudonym of user at SPB,
    ...
    encrypted pseudonym of user at SPN ]

The IdP and IM may include attribute data in the tokens they emit. This attribute data can be kept in any suitable data structure, usually indexed by user and sometimes by SP, or both.

The IM needs additional data structure to determine what services are available to a User. In its simplest form this would consist of

User’s pseudonym Service Type SP EntityID
---------------- ----------- ------------
789IM    Role Author. C.example.com
789IM    HR Authority B.example.com
579IM    Role Author. C.example.com
579IM    HR Authority B.example.com

but other more general realizations can include data needed for Trust and Privacy Negotiation phase of Discovery. These will be explored in the Trust and Privacy Negotiation documentation.
An IDP may have a limited form of this table to cover the necessity of emitting IM bootstrap token during SSO.

All parties - IDP, IM, and SP (FE or WS) - need to maintain some metadata about each other. Such metadata may include SOAP endpoints, protocol profiles and bindings to use, etc. These will generally be specified in protocol specific documents as adopted in [TAS3PROTO], but for general idea the reader may want to see [SAML2meta].

There is also the requirement for a user to be able to aggregate his attributes together in order to gain access to web services. This requires an attribute linking service, which is fully described in [TAS3D71IdMAnAz].

2.4.2 Personal Data and Applications

A SP can use whatever data model it desires (TAS3 Architecture is not prescriptive in this regard) in storing the data about the Users as long as it meets security and privacy guarantees detailed in [TAS3COMPLIANCE]. The persistent pseudonym of the User suggests an obvious database key, but other arrangements are possible.

TAS3 Architecture foresees aggregation of data from multiple sources with its support for policy aggregation. One common realization of this approach is to consider a document as a collection of external data streams, please see [TAS3D81RepoSW]. This approach will be supported by some of the TAS3 software deliverables (e.g. output of WP8).

2.4.3 Using Sticky Policies to Protect Data

Sticky policies can be attached to most data items and are especially foreseen to protect personal data and control its dissemination. The purpose for which the data was collected is expressed as sticky policy. This section addresses Reqs. D1.2-2.21-DataProtLaw, D1.2-6.5-Purpose, and D1.2-4.1-EnfUCPol. Data origin and collection method can also be indicated using sticky policies (Req. D1.2-6.8-UserAccess).

Sticky policies are evaluated as part of the authorization process. They should ideally be bound to the data they protect by encryption and signing solution that would prevent disclosure of the data unless the policy evaluates to permit. However, this is a difficult research problem and will be addressed in other TAS3 deliverables.

2.4.4 Using Encryption to Protect Data

All protocol flows use encryption. Usually this will be in form of connection level encryption, but in certain cases application layer public key encryption will be used to protect tokens or attribute data while it is in transit through an intermediary (e.g. IM token when passing through FE).

2.5 Authorization Process

This section partially addresses Req. D1.2-6.12-Sec.

Fig-2.8 depicts refined structure of the Authorization Process.

1. Central notion is that the Web Service PEP ("=" above the WS1 in the figure) calls a Master PDP, which then gathers the authorization from whatever sources it can.

2. Some of the data used for the decision may have come from the Web Service itself (it may have been inline in the Request, or the Web Service may know it otherwise), but if additional data is needed, the Master PDP will contact Policy Information Points (PIPs) as appropriate. (Processing of PIP request itself is an instance of Enforcement and Authorization Process, thus giving all of this rather recursive flavour.)

3. Trust and/or Reputation may be a factor in the authorization decision. This is handled by modelling the Trust and Reputation Provider (labelled as just "Trust" in the figure) as just another PDP that the Master PDP calls. The feedback and inputs to the Reputation computation are not shown here.
4. Given that sticky policies may potentially be written in different policy languages, the Master PDP will detect the language and call appropriate PDP to have the policy interpreted.

2.6 Enforcement Process

This section partially addresses Req. D1.2-6.12-Sec.

When a Web Services call is made, there are several control points in the flow, as shown in Fig. 2.10.

1. Request is first controlled by a Requester, i.e. on Client side, for being an acceptable request. For example, if the request is about to submit data to the Service Provider, there may be several policies about what can be submitted.

2. The controls can have multiple facets, i.e. the application programmer may have programmed in some implicit policies, the organization that operates the application may have some policies of its own, the Trust Network is certain to have policies, and finally the User himself may have set up some policies (which may involve attaching sticky policies to the data). Conceptually these are addressed by a PEP contacting Master PDP which may contact stake holder specific PDPs. If different stake holder policies result in a conflict, the Master PDP implements a Conflict Resolution Policy to arrive at a decision. An alternative approach is to use Identity Governance Framework [IGF] CARML declaration to set up the PEP, or some part of it.

3. After request has been authorized to send, the Service Provider will examine if the request is acceptable using a similar stack of PEPs. Examination on Service Provider side is the "traditional" enforcement point that most people think about. It filters out inappropriate data requests as well as illicit writes.

4. When preparing to ship response, the Service Provider uses a PEP and Master PDP to further filter the response. Although the request side PEP should have made sure that only legitimate requests can ever...
5. When Client receives the response, it examines it with a PEP and Master PDP. Such examination may be necessary to understand if there were sticky policies attached, or to perform obligations. Given the rules under which the Service Provider released the data, it may be that Client finds that it can not use...
the data for the intended purpose and therefore has to reject the request.

6. Not depicted, but logically part of the Client Request sending side are also
   a. discovery
   b. trust negotiation and establishment
   c. signing of request

7. Not depicted, but logically part of the Service Provider Request processing are also
   a. trust negotiation and establishment
   b. validation of message structure
   c. signature validation

8. Not depicted, but logically part of the Service Provider Response sending side are also
   a. signing of response

9. Not depicted, but logically part of the Client Response reception side are also
   a. validation of message structure
   b. validation of correlation
   c. signature validation
   d. processing obligations

2.7 Configuration Process

TAS3 is pervasively model driven. Fig-2.11 shows how a business process model can drive auditing processes, or even influence the Dashboard user interface so that Users can visualize the processes.

Fig-2.9, shows how models are used to configure policies for the PDPs. It also shows an alternate approach where PEP itself can be directly configured, e.g. using Identity Governance Framework [IGF] CARML and/or AAPML.

From the model the Trust Guarantor is able to derive

- Basic trust configuration, i.e. who belongs to the network
  - a white list of members
  - metadata to configure trust in the members

- Configurations to be pushed to operational elements of the network so that they will consistently enforce the process and trust model and the security model of the Trust Network.

- Operations Monitoring setup, e.g. if alerts are coming from some node, what Networks Operations Centre (NOC) process should they enter, where should they be disseminated, who should see them, and who is responsible for response

- On-line compliance testing configuration. This will drive a robot, spider if you like, that will comb through the Trust Network on a regular basis to verify that each service is in compliance with the policies that it publicly manifests.

The Organizations A and B participate in the Trust Network. They also model their business processes, extending and refining the global model. They, too, will benefit from ability to automatically configure and monitor the components of their infrastructure.
Figure 2.11: Pushing configurations from model.

2.8 Audit

No central audit log. There will not be any central audit log. Only audit data released routinely out of an organization or Service Provider are references to audit events and anonymized summary data. If an audit needs to drill into the audit trail, the authorized auditor will be given access, upon escalation, to fetch or view the local audit trails and ability to correlate the events to form a “big picture”. Without such authorization correlation will not be possible. This principle applies to the User’s Dashboard as well.

The audit domain is essential to maintain the validity of the trust fabric in the infrastructure. The domain will receive data on authorisation decision as illustrated in Fig-2.13. This enables the domain to become a central point for monitoring of authorisation processes in individual TAS3 instances.

The services in the auditing and monitoring domain will receive other forms of data linked to trust from the TAS3 infrastructure. This data will also include information on service invocations and workflow execution. The data from the results of these events will be stored in two main sets of services in the auditing and monitoring domain, there are auditing and compliance tools and operation monitoring tools.

It is important to note that these two sets of information will be handled quite separately. The operation monitoring tools will be operated by the applications code and will be application specific, whereas the auditing and monitoring will be operated by the TAS3 security layer and will be application independent.

The data collected from the monitoring in the audits can be then used by elements in the infrastructure such as the Dashboard. This will enable users to look at both how their data has been used in the infrastructure and also if any services have failed in this execution. In cases of failure or rogue behaviour the negative feedback from this can be fed to the Trust and Reputation service.
Some Audit Principles:

- Nobody should be able to tamper with the audit trail without detection. This includes insertion or removal of audit records, altering of audit records and deletion of entire audit files.
- Nobody should be able to put together the entire audit trail without proper authorization
- When answering a user audit request, the initial answer may have coarse granularity, such as organizations that have accessed. Only upon more thorough, authorized, investigation more detail, such as employees that have accessed would be revealed.

Relevant prior art will be incorporated in a future version of this document including regulatory compliance and best practises from

- Directive 95/46/EC and other guarantees offered by EU wide data protection legislation
- SOX [SOX02]
- SAS70
- COBIT
- ISO/IEC 27001
- Liberty Alliance
Figure 2.13: Auditing an authorization decision.

- Identity Assurance Framework [IAF]
- Identity Governance Framework [IGF] Privacy Constraints

- COSO
- PCI DSS [PCI08]
Figure 2.14: Monitoring operation of the network using the configured model.
3 Core Security Architecture

This section specifies much of the logistics that allow the identity of the user to be passed around between the architectural entities. This is a nontrivial problem, especially if pseudonymous delegated identity is to be supported, combined with recursive calls.

3.1 Flows

This section addresses Reqs. D1.2-2.14-Priv, D1.2-2.15-Resp, D1.2-2.18-AnCredi, D1.2-2.19-AzCredi, D1.2-2.20-Az, D1.2-6.12-Sec, D1.2-6.17-TechBind, D1.2-7.3-An, D1.2-7.8-NoColl, D1.2-7.16-Nym, D1.2-7.21-Safe, D1.2-4.2-BPPrivacy, D1.2-4.4-CourtProof.

![Diagram: General detailed flow of a service request](image)

**Figure 3.1:** General detailed flow of a service request

1. A client application wishing to call some service in another organization, initiates the call.
2. The Client PEP will enforce outbound authorization decision. To be able to do this, it first engages in Trust and Privacy Negotiation, which is a discovery process, see Section 3.6, and then forwards the request to the web services stack.
3. Web services Stack (the "Stack") will compose a request message including the identity tokens that are needed and signs the message. It then send the message to the Stack on the service side.
4. The service Stack will authenticate the sending Stack and verify the digital signature. The acceptance of the message will depend on a degree of trust on the signing party, which was established during the Trust and Privacy Negotiation.
5. The service inbound PEP will consult the Master PDP to determine if the service request should be allowed to go forward.
6. The inbound PEP will pass the request to the payload service, which will reply.

7. The outbound PEP of the Service will validate that the data can be released and attach obligations.

8. The Stack at the service correlates the response to the request, signs it and sends the response.

9. The client Stack receives the response, checks its correlation with the request, and verifies the signatures in the response.

10. The client inbound PEP checks that the response is authorized and complies with the obligations that were received.

11. The payload message is passed to the Client Application.

### 3.2 Tokens, Access Credentials

A central problem in multi-tier (or recursive) web services architecture is propagation of identity, or identity handle, to all tiers, while preserving privacy separation (resilience to collusion) between the parties.

The identity handle can allow, if chosen, linking of user’s consecutive visits together so that the service can collect data about the user for future reference and provision of the service. In this case the user is persistently identified, but to preserve privacy, the user will be identified differently towards different parties. This prevents collusion by the parties.

Sometimes it is undesirable for the service to link relate visits of the user together. In this case user is identified transiently, i.e. by one-time pseudorandom identifier (Req. D1.2-7.18-Seq). Within one overall session, user can be identified persistently towards one service while at the same time transiently towards another service.

In general access credentials come in the form of tokens that are digitally signed by a system entity, usually a Trusted Third Party, such as an IdP or ID Mapper service. Reader can use SAML assertion [SAML2core] as a mental model, though this is not the only possible technology choice.

This section addresses Reqs. D1.2-7.4-MultiCred and D1.2-7.18-Seq.

#### 3.2.1 Attribute Pull Model

**Target model.** Fully capable. All use cases work and best privacy properties. This model has been extensively studied in Liberty Alliance standardization work (n.b. this does not limit its applicability to Liberty ID-WSF - same concept can be implemented using other web service specifications, albeit with lesser maturity). This model addresses minimal disclosure particularly well, thus contributing to satisfying Reqs. D1.2-2.14-Priv D1.2-2.21-DataProtLaw, D1.2-6.4-Min, D1.2-7.5-Min, D1.2-7.12-CredStepUp, D1.2-6.86-Min, D1.2-9.1-SecData.

The Pull Model consists of front channel SSO layer and back channel web services layer. The "pull" refers to the strategy where attributes are requested from their authoritative sources only on as needed basis.

This has several benefits:

1. Minimal disclosure - only needed attributes are generated and shipped
2. Direct relationship with Attribute Authority. No intermediaries which could gain undue knowledge. This may also reduce crypto overhead as protection against the in-transit man-in-the-middle is not needed.
3. Intermediaries do not need to guess what attributes might be needed down the web services call chain or in a particular variant of a business process.
4. Fully dynamic and recursive operation that supports several Business Process Topologies. At least all forms of sequence (horizontal or vertical) and trees are supported. Support for a DAG would seem feasible. Other topologies need further study.
Use Case  User U authenticates with a service provider A through her IdP1. A needs to invoke further service providers with reference to U.

Problem Definition If the trusted architecture uses a unique, even if random and transitory, userID throughout then such a userID would allow multiple parties to collude and correlate all data belonging to U.

Objective The system must avoid producing correlation handles in the process.

Solution Idea Each service provider knows the user by a different random userID, a persistent pseudonym. And these pseudonyms are held by a mapping service. When one service provider wants to pass on the request to another service provider, it can ask mapping service for a lookup of the pseudonymous userID in the target service provider.

Given that the user’s pseudonym at the other provider is encrypted in transit, this solution avoids any service providers sharing correlation handles. (N.B. In this system the two service providers invoking each other’s services may still be able to directly collude, see Threat T107-LogTokLeak, but the service providers at the ends of a chain of services where chain length > 2 can not collude. The solution is to not log the tokens, see CR53-DontLogTok.)

3.2.1.1 Front Channel

Trivial situation is when the payload application consists entirely of a web gui or web site, without any web services call. Never-the-less, this is a very important flow because it is the most common way for Users to interact with the system. It is also a necessary precondition for the web services flows to be initiated and bootstrapped with the necessary tokens, including the IM access token.

![Flow at front channel](image)

Figure 3.2: Flow at front channel

Example: In our concrete example U authenticates and makes a service request to A which invokes...
another service provider B which also contains information about user U.

Assumptions:

- IdP has a table that lists the user name and the password of the user.
- IdP passes the permanent userID with a given Service Provider to that Service Provider every time the user logs in to the IdP from the Service Provider. The identifier is conveyed in a token, e.g. `<username: sampo; attribute: member of tas3; other: permanent userID of user with different service providers>`

The Steps of the Protocol with one layer of Service Provider Invocations

1. User U wants to access service provider A and starts interaction with A.
2. U is redirected to IdP1 (n.b. IdP discovery is not addressed in this flow, though industry standards like [SAML2core] and [CardSpace] do address it)
3. U logs in at IdP1. The authentication method is out-of-scope for this flow.
   IdP1 returns two encrypted tokens to A:
   - **TokenAuthn** the token contains U’s permanent pseudonymous userID 123A4. It is encrypted such that only A can read it and authenticate the user.
   - **TokenIM** the token is encrypted for the IM and contains the permanent pseudonymous userID of U with IM which is 789IM. The token is bound to A (contains an indication that only A can use it towards IM).

A authenticates U with TokenAuthn (possibly Single Sign-On - SSO). If it is a stand alone service, A returns the results of the services to U and A is done.

3.2.1.2 Front Channel Using Identity Selector

Identity Selector technology aims at solving the IdP selection problem. The central proposal in the area is InfoCard, which is realized by Microsoft CardSpace and some open source Identity Selectors. InfoCard can be deployed in direct fashion, but the problem has been availability of SAML 2.0 tokens. This is usually solved by deploying InfoCard in a proxy setup, as shown in Fig-3.3.

3.2.1.3 Back Channel, Simple

This flow expands on front channel by adding one web services call on back channel. This section addresses Req. *D1.2.3.10-JITPerm*.

Example: In our concrete example U authenticates and makes a service request to A which invokes another service provider B which also contains information about user U.

Assumptions:

- There is service provider IdMapper (IM). Each user usually has one IM that knows the permanent userID at the different service providers.
- IdPs know the IMs of the users (there are several ways to know. See section on user registration in this document to be written.)
- IdP/IM produce IM tokens. The IM tokens include the following information (which means this information is known to the IdP s and IMs):
The Steps of the Protocol with one layer of Service Provider Invocations

1. (Same as above.) User U wants to access service provider A and starts interaction with A.

2. (Same as above.) U is redirected to IdP1 (n.b. IdP discovery is not addressed in this flow, though industry standards like [SAML2core] and [CardSpace] do address it)

3. (Same as above.) U logs in at IdP1. The authentication method is out-of-scope for this flow.

IdP1 returns two encrypted tokens to A:

- **TokenAuthn** the token contains U’s permanent pseudonymous userID 123A4. It is encrypted such that only A can read it and authenticate the user.

- **TokenIM** the token is encrypted for the IM and contains the permanent pseudonymous userID of U with IM which is 789IM. The token is bound to A (contains an indication that only A can use it towards IM).

A authenticates U with TokenAuthn (possibly Single Sign-On - SSO).
4. A needs to use other service provider B to complete the services and needs the permanent pseudonymous userID for U in B. For this A passes TokenIM from IdP1 to IM.

The service provider B is selected based on Trust and Privacy Negotiator’s efforts to find a suitably trusted SP from the database maintained by the IM (or some other part of the Discovery functionality).

(Req. D1.2-3.10-JITPerm)

IM decrypts TokenIM from IdP1 and sees the user U registered as 789IM in its database. This with the token serves to authenticate the user to IM. Provided that the expiration time of the token is relatively short, the user can be assumed to be present (User Present scenario).

B looks up the userID of 789IM for service provider B which is 456B (the lookup values can be in encrypted form).

5. IM encrypts two new tokens for the invoked service providers B and gives them to A.

TokenUIDinB the token is encrypted for B. The token contains the pseudonymous identity of user U at B. In this case it is: 456B.

TokenIM the token is encrypted for the IM and contains the userID of U with IM which is 789IM The token is bound to B.

6. A sends a request to B for a service for U and sends the two tokens from the IM.

B decrypts the token and recognizes the user as having UserID 456B in its database.

B sees that 456B is the user U. It calls the authorization function to see if U is authorized. Assuming the answer is granted, the service is provided.

If B needs to invoke further services with a service provider C it communicates with the IM of U using its TokenIM and repeats the steps 4 through 6. See the recursive case, below.
7. B returns a result to A which completes the service and returns result to user U.

If a User has multiple IMs, multiple IM tokens would be generated if there was no way to ask User’s choice or other deciding rule to pick just one. This may result in practice nearly all IMs being aware of each other, but this need not always be the case and even partially populated IM matrix would remain useful to the user. Further, the IM matrix may be different for different users.

3.2.1.4 IM Bootstrap Token Minting and Passing through Front Channel

A key complication in the operation of the back channel is how to get the ball rolling, i.e. where do the first tokens come, before we can discover more tokens. The simple idea of just using the front channel token has undesirable privacy ramifications as it would provide a correlation handle between the SP and the discovery.

Such correlation handle can be avoided by bootstrapping procedure where the IdP provides a separate, encrypted, token for access to the discovery. Although SP will be an intermediary in passing the token to the discovery, it can not learn a correlation handle due to the encryption. Consider Fig-3.5 where the Single Sign-On (SSO) assertion (a7n), shown as red oval, is minted by the IdP, with another assertion, the discovery bootstrap token shown as blue ball, in it. The SP will establish session for the User (Principal) using the SSO assertion. When it needs to call a web service, it will extract the bootstrap token and pass it to the discovery.

One might ask how does the discovery know all the services the user has and what identity to include in the token. Many methods are possible, but ultimately the discovery maintains a federation database of pseudonyms at each web service for the user. This is very similar to what IdP maintains and it is not uncommon for IdP and discovery to be operated by the same organization.

One way to create the database is to bulk provision it. Other way is to have user’s actively register the services they consider theirs. Consider Fig-3.6 where user first (1) visits a service, performing a Single Sign-On, thus establishing his pseudonymous identity at the service. Then (2) user triggers the service to register itself as one of the user’s services. At this point the discovery database records what it should send as users identity in a subsequent web service call. When the call is made, first the discovery step (4) is made to obtain the token and then (5) the actual web service call with the correct identity.

3.2.1.5 Improvement Idea: Late IM Token Request

N.B. The IM does not get used at the last step of the chain. It has to produce n + 1 tokens for n invocations. This introduces a slight inefficiency.

An improvement of the efficiency of the process is as follows:
Each service provider is only given the authn token and is not given the IM token. If the service provider can provide the service then no IM token is needed. If the service provider needs to contact another service provider, then it contacts the IM to ask for the ID of the user at the next service provider. It refers to the user using the permanent ID by which the user is known to the IM and B (e.g. 456B for B or 123A for A). In this case 789IM is never known to any of the service providers and is internal to the IM. The IM can use the permanent ID to look up the user, find its local ID (789) then locate the permanent ID at the next service provider and send this encrypted for the next service provider, back to the requesting service provider for it to forward to the new service provider.

Problems with this approach: if there are multiple IMs, the service provider will not know which one to contact. If there is only one IM, this is ok. But, the protocol is not standard. Where as the protocol we defined above is a standard Liberty Alliance protocol.

### 3.2.1.6 Back Channel, Recursive

The Steps of the Protocol with one layer of Service Provider Invocations

1. (Same as above.) User U wants to access service provider A and starts interaction with A.
2. (Same as above.) U is redirected to IdP1 (n.b. IdP discovery is not addressed in this flow, though industry standards like [SAML2core] and [CardSpace] do address it)
3. (Same as above.) U logs in at IdP1. The authentication method is out-of-scope for this flow.
   
   IfP returns two encrypted tokens to A:

   **TokenAuthn** the token contains U’s permanent pseudonymous userID 123A4. It is encrypted such that only A can read it and authenticate the user.

   **TokenIM** the token is encrypted for the IM and contains the permanent pseudonymous userID of U with IM which is 789IM. The token is bound to A (contains an indication that only A can use it towards IM).

   A authenticates U with TokenAuthn (possibly Single Sign-On - SSO).
4. A needs to use other service provider B to complete the services and needs the permanent pseudonymous userID for U in B. For this A passes TokenIM from IdP1 to IM.

The service provider B is selected based on Trust and Privacy Negotiator’s efforts to find a suitably trusted SP from the database maintained by the IM (or some other part of the Discovery functionality).
IM decrypts TokenIM from IdP1 and sees the user U registered as 789IM in its database. This with the token serves to authenticate the user to IM. Provided that the expiration time of the token is relatively short, the user can be assumed to be present (User Present scenario).

B looks up the userID of 789IM for service provider B which is 456B (the lookup values can be in encrypted form).

5. IM encrypts two new tokens for the invoked service providers B and gives them to A.

- **TokenUIDinB**: the token is encrypted for B. The token contains the pseudonymous identity of user U at B. In this case it is: 456B.
- **TokenIM**: the token is encrypted for the IM and contains the userID of U with IM which is 789IM. The token is bound to B.

6. A sends a request to B for a service for U and sends the two tokens from the IM.

B decrypts the token and recognizes the user as having UserID 456B in its database.

B sees that 456B is the user U. It calls the authorization function to see if U is authorized. Assuming the answer is granted, the service is provided.

7. In course of providing the service, B wishes to call C. This is termed "recursive call" and such pattern can occur to any depth. B starts by discovering service of type "Role Authority", sending the IM token to the Identity Mapper.

8. The Identity Mapper decrypts the IM token and recovers the pseudonymous persistent ID 789IM, which is then used to locate from the database of IM the pseudonym of the User at service C, which is the only service of type "Role Authority" registered for the User. Identity Mapper returns two tokens: (i) the pseudonym of user at C encrypted such that only C can open it ("E(fgh)C" in figure), and (ii) IM token that C may use to make further web services calls.

9. B calls C, passing the tokens along.

10. C decrypts the token "E(fgh)C" and recovers the persistent pseudonym "fgh". It uses this key to look up the role from the database and returns it to B.

11. B uses the role to authorize the request (6) and returns a result to A which completes the service and returns result to user U.

Steps 7 through 10 can be repeated any number of times in a recursive fashion.

### 3.2.2 Linking Service: Attribute Push Model

This section addresses Req. D1.2-7.15-PushCred.

The Linking Service model is described more fully in [TAS3D71IdMAnAz]. We just give a brief summary of the model here.

The Linking Service model is based on the following assumptions/requirements:

- users typically have multiple attributes (authorisation credentials) assigned by multiple authorities and they are known by different identifiers at each authority
- some service providers will require many of these credentials in order to grant access
- the user does not want the inconvenience of having to authenticate (login) to each of the attribute authority in order to obtain credentials to give to the service provider
- the service provider does want strong cryptographic evidence that each of the authorisation credentials does belong to the user who has initiated the session
Figure 3.8: Linking Service: Registration phase.

- the user should be able to set up his policy for which attributes are aggregated by which SPs
- the user should be able to provide consent each time his attributes are aggregated by an SP.

The Linking Service is a new component that is under the control of the user, and allows the user to set his link release policy for which of his IdPs may be linked together so that their attributes can be aggregated and sent to the same SP. The user may in addition set an attribute release policy at each of his IdPs that is authoritative for more than one attribute, to say which SP can receive which subset of his attributes from this IdP. Taken together, the link release policy and the set of attribute release policies give the user complete control over which of his attributes can be aggregated together and released to which service providers. The GUI for the link release policy has already been described in Section ?? . The GUI for the attribute release policy will be very similar to this, but instead of associating IdPs with SPs, the user will associate attributes with SPs. Note that in many cases attribute release policies will not be needed since most IdPs are typically only authoritative for one attribute.

Once the user has linked his IdPs together and set his link release policy at the Linking Service, the user contacts an SP for a service. The front channel steps are as follows

1. User contacts SP asking for a Service
2. SP and/or user interact, allowing IdP choice as usual
3. User is redirected to chosen IdP and Authentication with the IdP takes place as usual
4. In addition the User can tick a box giving consent for attribute aggregation to take place in this session (see Fig.3.10)
1. User makes a service request.  2. User is redirected to her chosen IdP  3. User authenticates to IdP 1.  4. IdP 1 returns an authentication statement + attribute assertions + referral to linking service  5. SP follows referral  6. Linking service looks up IDP 1:PID 1 of user and finds links to other IdPs.  7. Linking service requests attributes from linked IdPs using respective PIDs  8. IdPs return signed and encrypted (to SP) attribute assertions.  9. Linking service relays all attribute assertions to SP.

Figure 3.9: Linking Service: Login with attribute push phase.

5. User is redirected back to SP and is granted access to the service.

Welcome. Login please.

Username: 
Password: 
☐ Do you wish to aggregate attributes from other linked accounts?
Login

Figure 3.10: The enhanced login screen for attribute aggregation.

The back channel communications that take place between steps 4 and 5 above are as follows:

1. The IdP sends an authentication SSO statement to the SP, containing a random identifier for the user. This prevents the SP from correlating the user’s requests in different sessions. (Note that if the user wishes to be correlated he can arrange for the linking service to send a unique identifier attribute from one of his attribute authorities during the attribute aggregation process, for example, a National Health Number from the Health Authority if the SP is a medical application, or a unique ID from the authenticating IdP). The IdP also sends an attribute statement containing the user’s attributes at this IdP, and an attribute statement containing referral(s) to the user’s linking service(s). Each referral contains the unique ID of the user as known by the Linking Service and it is encrypted to the public key of the Linking Service.

2. The SP acts on the referral(s) and contacts the LS’s discovery service asking it to return the linked IdPs’ discovery services, along with a Boolean saying “I will do it or You do it for me”.

TAS3_D2p1_Arch-v-23 (1.118)  Page 48 of 141
3. The LS decrypts the unique ID of the user, looks this up in its database and finds all the user’s linked accounts. If the SP has asked to perform aggregation itself, the linking service returns a Response containing referrals to the discovery services of the user’s linked IdPs.

4. The SP now sends a query message to each of the IdP’s discovery services, requesting the contact details of the user’s attribute authority. Alternatively, if the linking service is performing the aggregation on behalf of the SP, it sends the same message to each IdP.

5. The IdP’s discovery service locates the user’s local account by decrypting the user’s unique ID in the referral, and maps the random identifier from the authentication assertion into the user’s local account id. The IdP returns a Response containing the contact details of the attribute authority where the random identifier is now valid.

6. The SP or linking service sends an Attribute Query message to the attribute authority, using the random identifier, whereupon the attribute authority returns a digitally signed attribute assertion encrypted so that only the SP can read it.

7. If the linking service is doing the aggregation, it collects together all the encrypted responses from all the IdPs and then forwards the complete package to the SP.

8. The SP now has the following digitally signed assertions:

   a. An authentication assertion from a trusted IdP saying that the user has been authenticated, and is to be known by this random id for this session
   b. A set of attribute assertions from trusted attribute authorities saying that the user known by this random id possesses this set of attributes
   c. Based on the above the SP can authorize the user to access the requested resources, sure in the knowledge that trusted authorities have both authenticated the user and assigned attributes to her.

   3.2.2.1 N-Tier Linking Service Model

   This section addresses Req. D1.2-7.1-Deleg.

   If the SP above needs to subcontract one or more tasks to a backend service, and that backend service is to act from an authorization perspective as if the user herself had contacted it directly, then the backend service will need to be given the user’s authorization attributes. The exact model for how this is to be done is for further study in the following years of the project. Whilst there have been many previous models for dynamic delegation of authority none of them to the best of our knowledge have supported attribute based delegation simultaneously with privacy protection of the delegator’s and delegate’s identities. The following models at least are suitable for further study:

   A. Trusted SP. The first SP simply forwards the initial authentication statement and referrals from the authenticating IdP to the backend SP, and the backend SP follows exactly the same process as the first SP (described above). (Note that the attribute statements cannot be forwarded as these were targeted specifically for the first SP and encrypted to it.) The disadvantages of this method are:

      a. the backend service must trust that the first SP is authorised to forward the user’s authentication statement to it. Otherwise a rogue SP could illegally forward the authentication statement to a backend SP in order to defraud the user;
      b. the user either has to know beforehand which backend services are going to be contacted, so that she can proactively sets up specific Link Release Policies for these backend SPs, or if she does not know or is unaware that backend services are to be involved, she has to set up a default policy for All Other Services (as described in Section ??). Otherwise the backend SP is likely to deny access to the subcontracted task.
B. Direct Delegation. The user contacts a delegation service and delegates her attributes to the first SP, bestowing these credentials with delegation rights. The first SP uses these credentials to authorize the user, and then delegates them further to the backend SP, optionally bestowing further delegation rights on the backend SP in case it needs to subcontract further. The advantage of this model is that the backend SP does not need to trust the first SP as much, since the latter has specifically been delegated rights to it by the user. Further research is still needed here, since it is currently not known precisely how to delegate an aggregated set of attributes issued by multiple attribute authorities when the user is known by different identifiers at each authority. We will need to find a way to combine the linking and delegation services to work together in a trustworthy manner.

C. FileSpace. We use a completely different model for multiple credential authorisation, termed FileSpace [Chadwick09]. This gives the user a set of files which he can copy freely between devices and service providers. Service providers can also freely copy the files between each other to delegate authority on behalf of the user. The files are actually encrypted authorization credentials signed by their respective attribute authorities. Consequently they are useless to the recipient (who can be a thief or a genuine SP) until it gets the decryption key. Herein lies the twist. The user is the only person with the private key that can decrypt them. Thus before any front end or backend SP can utilise the credentials they must ask the user for the decryption key. Once they have the decryption key they know that (a) the user is the genuine owner of the credentials as she was able to decrypt them, (b) the attributes genuinely belong to the user because they are signed by a trusted attribute authority, and (c) the user has given consent for them to be used (because she provided the decryption key). If we place the user’s private key in the Dashboard, then each SP that wants to authorize the user must first contact the Dashboard asking for a decryption key and the user can keep track of progress of his task as various events arrive at the Dashboard. She can then give consent (or not) as appropriate.

D. Shibboleth Portal. The Shibboleth community has developed a delegation model, initially targeted at web portals and portlets, but generalized so that it can be used for any web services based delegation. It is based on the Liberty Alliance ID-WSF Enhanced Client or Proxy SSO Profile [SOAPAuthn2] instead of the SAML 2.0 Web Browser SSO Profile [SAML2prof] which Shibboleth currently uses for SSO. It works by the first SP going back to the user’s IdP to authenticate itself and ask for a new authentication statement allowing it to become a delegate of the user and a delegator to a backend SP. The user’s initial authentication exchange with the IdP is enhanced to allow the user to specifically authorize delegation by the SP it is contacting. This causes the IdP to insert a new token into the authentication statement which authorizes the recipient (the SP) to call it back to become a delegate. After the SP has authenticated and authorized the user, the SP contacts a backend SP and quite naturally is required to authenticate to the latter, as is any user. So the SP then contacts the IdP, authenticates to it (say by using mutual TLS) and passes the token from the user’s authentication statement. This notifies the IdP that the SP is entitled to act as a delegate of the user, so it issues an authentication statement in the name of the original user, to the SP. This statement is targeted at the backend SP, and again contains a token that allows the backend SP to call it back to become a delegate of the user in future. The first SP passes the authentication statement to the backend SP and thereby masquerades as the user.

E. Subcontracting. Of course the first SP can always subcontract the backend SP to perform the task on its own behalf (as is typically done in manufacturing supply chains today), in which case the user’s attributes are not needed by any backend service.

3.2.3 Simple Attribute Push Model

Recommended approach for initial deployments that have not yet developed full infrastructure.

In this model some commonly needed, or "enabler", attributes such as Trust Network membership or role are supplied directly as part of Single Sign-On (SSO) or web service tokens. Other perhaps justifiable attributes, that do not provoke undue privacy or legal implications, could be
• legally nonbinding nickname for greeting user
• user’s preferred language

This model implies that IdP or IDMapper assume some of the responsibilities of an Attribute Authority. This is well supported in existing protocols and available software implementations. It is also probably the largest operation model in use today in existing federations. For example, this is the model used by all Shibboleth implementations such as the UK academic community federation which has over 800 IdPs and SPs since it was launched in August 2008. The number is still continuing to grow linearly with another 20 or so providers being added per month.

Drawbacks of this approach are

1. Only a very narrow set of attributes will be universally needed by nearly all Front Ends or Web Services.
2. Danger of nonadherence to minimal disclosure principles - its easy to have creep where "just one more" attribute is added to support "just one more" application. This is also wasteful in that cost for generating attribute statements that are seldom needed is still paid on every transaction.
   A solution to this is to have an Attribute Release Policy (ARP) at the IdP which provides rules for which attributes should be released to which SPs. In this way the attributes can be effectively filtered before release. The ARP is set by the user and/or the IdP itself, and open source software does exist for this. The design is very similar to the IdP Release Policy of the Linking Service described in Section 3.2.2, above. Still, this approach lacks granularity as the attribute needs of a SP are assumed to be always the same, while in reality SP may run various different business processes with different needs.
3. Postponement of moving to full pull model.

3.3 Delegation

This section addresses Reqs. D1.2-3.7-Deleg and D1.2-7.1-Deleg.

Technical clarification: Here "Delegation" means assignment of decision powers, and access rights they imply, from one User to another User. This is distinct from merely instructing some web application or service to act on ones behalf - some other practitioners call also this "delegation", but we find a service merely executing on instructions of a user so common that it is the base case, and does not need special mention.

Delegation is analogous to giving personal banker the right to manage your portfolio, while action on behalf happens when bank executes wire transfer upon you direct orders.

It should be noted that some system entities may be modelled as juridical persons and can, thus, participate in Delegation like the Users can.

General properties of delegation are

• Express and auditable act of creation (e.g. issue power of attorney), with indication of registration (where needed).
  - Specification of delegatee ("Performer" of a web service action)
  - Specification of delegator (sometimes, if a resource of the delegator is involved, "Target" of a web service action)
  - Specification of scope
    - Actions and/or
    - Resources
    - Role based delegation
- Specification of expiry and other policy constraints

- Sub-delegation, when this ability is expressly mentioned in constraints

- Ability to revoke, with extent to with delegation is possible, such as
  - At any step of sub-delegation chain
  - Any superior ancestor can revoke any descendant
  - Audit

- Divulgence of issuer’s signing private key MUST NOT be used as a mechanism of delegation.

- Expression of the delegation and target in web services calls at any level of recursion

- Verification by Relying Party: mandate assurance & authenticity (typically verify sig on token + query of MA for revocation info + evaluation of possible constraints)

- Transparency: ability for user to verify which mandates have in fact been exercised or formally accepted.

This could be implemented by the Delegation Service feeding information about each invitation usage to the Audit Event Bus, where the Dashboard can pick up the information and display it to the user when user comes to consult it. Also, when Service Responder, or its CVS or PDP, consumes a delegation token, it will inform the Audit Event Bus so that the Dashboard can have the big picture of the delegation usage.

Delegation is also discussed in section 6 “The Delegation Service” of [TAS3D42Repo] and in section 6 of Deliverable D7.1 [TAS3D71IdMAnAz].

Designation of Delegatee may be by

- Delegation to anyone (or first one) possessing an invitation token

- Delegation to someone specific
  - How is the someone identified? Well known unique ID is an option.

- Delegation to anyone having specific role

- Delegation to anyone having some relationship to the Delegator

- Delegation to anyone having some relationship to previous tasks, business process steps, or environmental context (e.g. physical access).

Steps of delegation

1. Identify delegate
2. OPTIONAL: Resolve invitation to delegatee (may involve additional authorization)
3. Authorize delegation, e.g. to check for separation of duties
4. Implement delegation, e.g.
   - Generate target identity token
   - Set up Az rules (consider special case)
   - Assign roles
   - Create relationship, such ownership (e.g. assign owner)
5. Terminate delegation

Use case 1 "Headhunter"

Headhunter Alice processes matching requests from customer. Each headhunter has assigned to it some customers and usually he would not handle requests from other customers. However, the headhunter can assign one of his cases to a colleague, Bob. In this case the colleague is able to process the request despite customer not being his. However, it may turn out that a headhunter Alice is sick and requests from his customers pile up. In this case the manager Charles is able to assign the tasks to other headhunters, such as David, without any intervention of the headhunter that owns the customer, Alice.

Use case 2 "Job seeker and a coach"

Job seeker arrives and starts job search. In the process job seeker needs help of a coach. A coach needs to be assigned. This can happen in many ways: use same coach job seeker has previously used, pick coach that is face to face with the job seeker, pick first coach that grabs the task, or assign any assigned and known coach.

3.3.1 Invitation Based Token Approach

The invitation based approach is modelled after the Liberty Alliance [1], [2].

Assume Alice has used ePortfolio Front End to construct some artifacts about her career (see Fig-3.11), including attestation from University A that she has a degree, and some exhibits, in Album A of the work she has already done.

Now, Alice can invite her job coach Bob to access these artifacts as Web Services, see Fig-3.12. First Bob resolves the invitation to the necessary access tokens and then accesses the services. Bob is acting through the Job Search Front End.

On access (2a) two tokens will be present

**Target Identity**  Delegation Token, encrypted for Album A and usable by Job Search, containing Alice’s pseudonym at Album A. This token indicates that the access is by invitation and not by Alice being present in the transaction. The Delegation token can also include description of which part of the user’s resource at the Service Provider can be accessed and how.

**Subject Identity**  Token, encrypted for Album A and usable by Job Search, containing Bob’s pseudonym
Figure 3.12: Bob using Alice’s Web Services by way of invitation.

at Album A. This token indicates that Bob is performing the operation and that Bob is present in the transaction (at front channel).
### 3.3.1.1 Details of the Invitation Flow

Consider Figs 3.13 and 3.14 and the depicted steps as follows:

1. **Delegator (Bob)** selects the resource to share at the Service Provider. Service Provider knows who Delegator is as Bob used Single Sign-On to login to the SP.

2. Once a resource has been selected, Delegator is transferred to the user interface of the Delegation Service. The Delegation Service generates an Invitation Ticket, which is formatted as URL pointing back to the Web GUI of the Delegation Service, but with a nonce component that is hard to guess. The invitation is remembered in the database of the Delegation Service.

3. The Delegator then gives the invitation to Delegatee. The Delegator does not need to know the specific identity of the Delegatee in the network. However, if the Delegator cares, he could use out of band...
means of ascertaining that the Delegatee that receives the invitation is the person to whom he wishes to delegate, e.g. instead of emailing the invitation, deliver it personally.

4. The Delegatee now resolves the invitation by clicking the URL and lands at the Web GUI of the Delegation Service, which asks the Delegatee to identify itself by means of an IdP (usually Single Sign-On). This IdP can be different from Delegator’s IdP as long as the Delegation Service and the SP are willing to trust it. Various mechanisms, that are described in Sections 3.7 and 3.6, to establish the trust exist.

If Delegatee does not have any IdP, then Delegatee should register with some IdP, otherwise he can not continue the flow.

N.B. A flow is possible where the invitation itself grants access to the resource without any need to authenticate the delegatee. While this flow may be interesting in some commercial settings, it lacks sufficient audit and accountability safe guards to be included in the TAS³ architecture.

5. Delegation Service invokes Delegatee’s Identity Mapping Service to convert the identity of the delegatee are the Delegation Service to the identity of the Delegatee at the SP and stores it in the invitation database. The identity will be encrypted so that only SP can open it.

6. The Delegation Service generates an artifact with nonce properties and associates it with the invitation. It then redirects the Delegatee to the user interface of the SP, passing the artifact in the query string.

7. SP dereferences the artifact by contacting on back channel the Delegation Service, which looks up the invitation using the artifact and generates a Delegation Token binding together the authorized resource (known from time when invitation was created) and the Delegatee (known from step 5); signs the token, and ships it to the SP.

8. The SP PEP now passes the invoking identity, i.e. the Delegatee, known from Single Sign-On and the (contents of) Delegation Token to the PDP, which will authorize the operation. Typically the authorization rule will require that the accessed resource and invocation identity match the Delegation Token.

In this flow, the Delegation Service and the SP could be merged, effectively optimizing steps 2 and 7 to function calls. While such merge is technically sound, it may hinder development of competitive market for the delegated services. The flows in [TAS3D42Repo] Appendix 2 “Proof of Ownership Using Permanent IDs” essentially presents the invitation flow where the invitation is then embedded in a composite document. That Appendix does not show how the Proof of Ownership URL (PoOURL) is dereferenced. Essentially the steps 4-8 could be performed, or other means to authorize the access could be used. It is important to understand that reliance on mere invitation does not leave audit trail trace about who accessed. The delegatee really needs to be identified (e.g. by SSO to Delegation Service) for the audit trail to be useful.

Essentially the steps 4-8, above, could be performed, or other means to authorize the access could be used. It is important to understand that reliance on mere invitation tokens (secret URLs) does not provide identity information to the audit trail about who accessed the document. But if authentication and authorisation of the PII recipient took place because a fixed proof public URL was used, then full audit information is provided.

3.3.1.2 Reuse of Invitation

One salient property of this flow is that the first time the invitation is dereferenced, it gets bound to a concrete user. In subsequent attempts to dereference the invitation a check can be made that it is the same user (but if requirement really is that the token should be reusable by different users, then this check can be omitted).
3.3.1.3 Application of Invitation Approach to Back Channel Web Service Calls

The invitations approach can be extended to cover back channel web service calls as well. This topic will be elaborated in the next version of this deliverable (D2.1 M30).

3.3.2 Mandate Tokens Approach

This approach is described in [Peeters09].
Next version of this deliverable (D2.1 M30) will outline application of Mandate Tokens to TAS³ architecture, including

- Emission
- Push Model
- Pull Model

### 3.3.3 Delegation by Direct Authorization Rule

In this model the resource owner, knowing delegatee’s full identification, creates an access control rule at the resource, i.e. sticky policy, that simply authorizes the delegatee to access the resource.

The ability to assign access to other user can be regulated by policies that are checked by the sticky policy interface, e.g. by consulting a PDP.

When delegatee accesses the resource, he identifies himself using the usual token passing flow, see Section 3.2, and the PDP is able to match his identity to the sticky policy authorizing the access.

**Difficulties are**

1. The Delegator has to have a solid identifier for the delegatee. This may be difficult for a human to know and accurately enter. It is possible to offer to the delegator a browsing or search interface of all potential delegatees, but such list may be unacceptable from data protection perspective and can be difficult to implement in a distributed way.

   A better alternative may be to adopt the invitation steps 1-4 from section 3.3.1 and modify the step 4 so that it edits the access control rule.

2. The resource has to have a sticky policy editing interface and access to this interface needs to be well controlled. Online editing of policies increases exposure and potential for compromise.

3. If policy editing interface is centralized (e.g. in Dashboard), the identification of the resource to which the policy applies may be difficult. This can be solved to an extent by Discovery. The policy editing interface of the resource would have to be web service based, with proof of presence of the delegator.

4. Privacy is poor as the requirement to know delegatee’s identity widely excludes pseudonymous approaches.

5. Invitations are not supported

### 3.3.4 Delegation by Role Based Authorization Rule

In this model the resource owner creates an access control rule at the resource, i.e. sticky policy, that authorizes anyone having a given role to access the resource.

The ability to assign access to a role can be regulated by policies that are checked by the sticky policy interface, e.g. by consulting a PDP.

The assignment of a role to a delegatee is performed by some role authority outside control (but still accountable through TAS³ oversight) of the resource owner. The usually role assignment is performed using a special Sharing GUI that the role authority accesses (his authorativeness is checked by separate access control policy verifying that he is a role authority). The role authority needs to identify the delegatee and then assign the role.

The role is stored in delegatee’s role repository, which can be his Attribute Provider.

When delegatee accesses the resource, he identifies himself using the usual token passing flow, see Section 3.2, and the PDP is able to retrieve his role and match that to the sticky policy authorizing the access. The role retrieval may be through push or pull model.
In its basic form the RBAC allows any role holder access to any resource that accepts the role and the identity of the role holder is irrelevant, thus permitting use of pseudonyms or transient identifiers that improve the privacy.

However the fact that the resource is not constrained is a problem. This could be solved by having as many roles as there are resources, but this would lead to combinatorial explosion in the number of roles and quickly become unworkable. Also, the role identifier itself could become a correlation handle across the resources of a user.

A way to constrain the role is to parametrize it. Parametrized roles have a main part that specifies the role proper and then a parameter that specifies to which resource the role applies. Since parametrized role identifier has unique identifier properties, it is highly liable to become a correlation handle.

This approach adds flexibility, but still suffers from need to have exposed policy editing interface and to some degree about the problem of identifying the delegatee and resource. In parametrized variant any privacy protection is quickly lost.

### 3.3.5 Token Based Delegation to Well Known Delegatee

If Delegatee’s accurate identification can be known, the delegator can instruct a Delegation Service to create a signed (by Delegation Service) token for accessing his resource. The fact that he can make this delegation is checked against issuing policies, such as “data owner can delegate access to it”.

The token can then be stored for future use in several places:

1. In Attribute Authority of the Resource Owner
2. In Attribute Provider of the Delegatee
3. In token store of the Delegation Service

When Delegatee accesses the resource, he can push the token, obtained from (2) or (3), to the web service that provides the resource. The token will identify the resource to which it applies, this is the Target Identity.

Sometimes the mere presence of the delegation token is sufficient for accessing the resource, this would be considered a bearer token.

However, in a more common case, the Delegatee also pushes a token identifying himself, e.g. a SSO token. This expresses the Subject Identity, i.e. the identity of the user performing the access. This allows the delegation token to be constrained to a user who can present token evidence of actually being the Delegatee. This is essentially a Holder-of-Key proof and produces strong audit trail that identifies who delegated and to whom.

Another variant is for the resource providing the web service to pull the delegation token from one of the sources. If Subject Identity is not known, but the accessed resource is known, then (1) can be used. In effect the discover of the Attribute Authority is keyed on the identity of the resource owner. Token can also be retrieved from (3). Finally, if the Subject Identity is known, pulling the token from (2) becomes possible.

### 3.3.6 Token Based Delegation of a Received Role

If a user has a role, given to it by some role authority, and policy permits delegation of this role, the user can go to the Sharing GUI and ask a delegation token to be issued with known delegatee and known target resource. Accurately identifying the delegatee and the target are serious problems as described above.

The ability to delegate only "received role" is expressed by Delegation Service having a policy which states that delegator can only delegate roles he has himself, as determined by the roles the authority has assigned to the user.

The issued token is then stored for use by method (2) or (3) as discussed earlier. The token can be used for access either by push or pull methods.
3.3.7 Multi-layer (Chained) Delegation

The token based delegation methods described above lend themselves to multi-layer delegation. In this case the delegation is with right to sub-delegate and the delegatee is able to request that further delegation tokens are created, or that an invitation is created.

For access authorization generally only the last step of a delegation chain is important, but for audit purposes the full chain is needed.

The flows and tokens associated with multi-layer delegation are a topic of research in the TAS3 project and will be further elaborated in a future version of this deliverable (D2.1 M30).

3.4 Subject of the PII Not Present -Transaction

There are two main categories for supporting PII-Subject-Not-Present transactions

1. PII Subject consented at some earlier time.

   If the consent was expressed by the PII Subject by creating a policy that authorizes the delegation, then we need to establish from the audit trail that only the user could have created the policy.

   If token based delegation is used, PII-Subject-Not-Present could be implemented by "cacheing" an access credential for long time. Due to obvious problems with long term valid access credential, we only allow cacheing the discovery bootstrap credential. Using this credential the WSC MUST request a fresh access credential for the PII-Subject-Not-Present transaction immediately before attempting the transaction. This ensures that the PII Subject has control and visibility since the Discovery will be a third party to the transaction, allowing additional audit correlation. Discovery can also be used as centralized revocation point for the consent to the off-line transaction up to the point when the transaction is actually made.

   The discovery token was originally issued for a purpose and when a transaction token is requested, including the PII-Subject-Not-Present situation, a purpose MUST be declared so that the authorization process at the discovery can make appropriate decisions.

2. Transaction is permitted by law or contract without consent of the PII Subject. In this case the big problem is that as the PII Subject might never have been present we need to consider how the Legitimate Initiator (LI) identifies the PII Subject in the first place.

   a. The PII Subject has a federated account at the Legitimate Initiator. In this case the LI can ask the IdP to issue a discovery token. Such issuance needs to be strictly controlled. The LI is first authenticated (e.g. using PKI at TLS level) and then LI presents the legitimate purpose why the token is sought. The IdP will consult a PDP which will make the policy decision whether under these circumstances the discovery token should be issued. The audit trail shall rigorously reflect these events. After the discovery token has been obtained, the rest of the steps are as in (1).

   b. The PII Subject has an account at the Legitimate Initiator, but it is not federated. The issuance step of (a) will have additional request to create the federation. If the PII Subject eventually ends up using the LI through SSO, the already created federation will be used.

   c. The PII Subject does not have an account anywhere. In this case a stand-in account needs to be created first and then the process of (b) and (a) will be followed. If the PII Subject eventually starts using the LI using SSO, a problem remains on how to associate the PII Subject to the already existing account. This may be possible by asking some identifying information, such as sector specific or national identifier upon first SSO use. If asking such information is infeasible, or the PII Subject can supply wrong information, we may well end up with user having two accounts at the LI. Depending on the circumstances, this may not be a problem, or an administrative procedure may be needed to combine the accounts.
3.5 Break-the-Glass Authorization

This section addresses Reqs. D1.2-3.9-BPRecover, D1.2-4.6-BrkGlass, and D1.2-7.22-BrkGlass. See [TAS3D71IdMAnAz] for further discussion of the Break-the-Glass authorization.

In a Break-the-Glass scenario an operation would not normally be authorized given the actor and the resource (including owner of the resource), but given justified and legitimate imperative need, the operation is authorized, usually with additional auditing enabled.

A classical example would be an unconscious patient brought to an emergency ward. The physician has imperative need to access the patient records, yet the patient can not consent to the access.

The Break-the-Glass authorization is an area of active current (2009) research and TAS 3 architecture will in its future versions incorporate the best innovations in this area. Our current approach is as follows:

- **PEP driven Break-the-Glass** (RECOMMENDED approach): The PEP attempts authorization which fails for normal access. At this point PEP determines if Break-the-Glass could be applicable (could be indicated by PDP or structurally known by the PEP). If so, the PEP may proceed to ask consent from the actor ("Do you want to invoke Break-the-Glass privileges and additional auditing?") using interaction facilities of the architecture, or in some cases may enable the Break-the-Glass automatically.

- Enabling Break-the-Glass forces additional audit messages to be generated at the PEP and by the service overall. It is also possible that the Break-the-Glass authorization is explicitly asked from the PDP in which case there will be audit entry also on the PDP side, permitting better cross correlation of the audit trails.

- **Policy editing driven Break-the-Glass**: In this approach, after initial denial of the authorization, some mechanism is invoked to modify the policy such that the same actor will succeed in obtaining authorization. This approach is NOT RECOMMENDED due to complex security implications of allowing policy editing.

- **Role driven Break-the-Glass**: This is similar to the Policy Editing approach, but the edit happens in the role authority. This approach is NOT RECOMMENDED for much the same reasons as Policy Editing: the role authority needs to be strictly controlled and introducing automated role editing mechanism is not desirable.

- **Delegation driven Break-the-Glass**: In this approach after the failure, the actor visits the delegation authority and claims Break-the-Glass situation. The delegation authority verifies according to its policies if the Break-the-Glass delegation is appropriate given the actors role, etc., the resource, and the context of the request. If acceptable, the Delegation Authority issues a delegation token with special field that indicates the Break-the-Glass nature of the delegation and records the fact to its audit trail. When the delegation token is wielded at the PEP, it recognizes the delegation authority as trustworthy and notices the Break-the-Glass indication, enabling more extensive logging. This approach can be made to work.

- **External Break-the-Glass IdP driven approach**: This approach is a front channel special case of the PEP driven approach with some features of the Delegation driven approach. The particularity is that the PEP redirects the user, with indication of the resource to be accessed, to a special IdP which is responsible for verifying (probably querying a PDP) whether the user is eligible for Break-the-Glass access. If so, it issues a token indicating that the user is eligible and has been granted access, what resource is to be accessed, and the fact that Break-the-Glass has been invoked. The PEP seeing this token grants access, but enables additional logging. The special IdP will also have audit trail of the events, so it will be possible to cross check the trails.

3.6 Trust and Privacy Negotiation

This section addresses Req. D1.2-7.17-Increm.

Trust and Privacy Negotiation logistics and flows are subject to TAS 3 research. A future version of this deliverable (D2.1 M30) will report on these mechanisms in detail.

To give rough overview, the Trust and Privacy Negotiation can be carried at two levels: on Front Channel a user interface (Web GUI of a Front End) will step user through sequence of mutually revealing more information until agreement can be reached and transaction can be completed. This may involve the
user agreeing to relax policies on his data, or the Service Provider agreeing to honour some additional user policy that it did not offer originally. The second level is Trust and Privacy Negotiation in the back channel. The Discovery Service plays a large role in the Trust and Privacy Negotiation at this level. The mechanics are further described in [TAS3D41ID].

3.7 Interoperation Across Trust Networks

General approach for interoperation across Trust Networks is described in [LibertyInterFed], which focuses on the token passing flows in inter-federation situation. In addition to token passing, interoperability at data level is needed, i.e. the ontologies in use in the different Trust Networks either need to be the same or they need to be mapped. In particular, authorization critical data needs to be mapped.

Next version of this deliverable (D2.1 M30) will provide specifics of interoparation, based on TAS<sup>3</sup> research and implementation experience.

3.7.1 Semantic Interoperability Engine

This section satisfied Reqs. D1.2-2.23-SemIOP, D1.2-3.14-PIIPolicyDisco, and D1.2-3.15-SecPreserve.

![Figure 3.15: Interoperation across contexts.](image)

A semantic interoperability provides different mechanisms to map ontological entities from heterogeneous entities. Castano et al. [Castano07] identify two main categories of ontology matching techniques; namely linguistic and contextual matching techniques. Linguistic techniques evaluate the similarity among ontological content (i.e. classes, roles and instances) based on their names or labels. The main characteristic of these techniques is that they evaluate the similarity between two strings of characters. For example, the edit distance counts the minimum number of changes, such as insertion, deletion and replacement of characters, required to transform one string into the other string [Levenshtein66]. Although linguistic techniques often provide highly precise mappings, these techniques tend to fail when there is little lexical overlap between the labels of ontological entities. Contextual matching techniques can remedy this problem by assuming that some of the meaning of an entity is conveyed by its context. For example, Dieng and Hug [Dieng98] calculate the similarity between two concepts based on their direct super-classes and/or direct subclasses and/or sibling classes. The semantic interoperability engine will include different algorithm of these types, and thus be able to deal with a wide range of mismatches. As a result, multiple
organisations using different conceptualisation will be able to interoperate to achieve a common goal.

As shown in Fig.3.15, the Semantic Interoperability Engine (SIE) works at conceptual level. It can be used process the two ontologies to produce Transforms that may be stored for later use. SIE can run as a scheduled batch to update the transforms, or change in either ontology can be used to trigger SIE. Finally it may be possible to invoke SIE dynamically whenever an existing transform is not yet available.

At runtime an efficient Transformer uses the Transforms to translate data in the messages that are exchanged. TAS³ focus is on using this mechanism to transform security, trust, and privacy related attributes such as roles. However, the mechanism in itself could be used for payload data transformation as well. The transform can be done in sending or receiving end. The Transformed has to be positioned such that each PEP (which calls PDP, passing the attributes along) will have the security related attributes in its own vocabulary. In the sending end this means after authorization, in receiving end before authorization.

### 3.8 Properties of Web Service Binding

Web Service Binding is a set of features that the communications layer is assumed to have. These features are often required by more sophisticated protection mechanisms like the token passing flows. They often address basic and well known threats like replay, unauthorized, and man-in-the-middle attacks in basic way while other mechanisms may address the same topics comprehensively, but in a more expensive way. Many of these features may seem selfevident, but we need to list them even if just to state the obvious.

1. Mutual authentication of the communicating entities MUST be possible. Usually this is done using transport layer digital certificates, but other approaches are possible.
2. Link confidentiality MUST be possible, usually using transport layer encryption.
3. Correlation
   - Request-Response Correlation
   - Business Process identification in correlation
4. Redirection support for flexibility
5. Recredentialing support (Req. D1.2-3.9-BPRecover)
6. Asynchronous support SHOULD be implemented (this will be addressed in a future version of this document)
7. Interaction Callback (or Exception Request)
   - Interaction Redirect (Req. D1.2-3.9-BPRecover)
   - Interaction Service (Req. D1.2-3.9-BPRecover)
8. Digital signing of messages for nonrepudiation (Reqs. D1.2-2.11-Transp, D1.2-2.15-Resp, D1.2-4.4-CourtProof)
9. Conveyance of Invoker and Target Identities, if web service uses identity.
4 Application Specific Architecture

4.1 Protocol Support for Conveyance of Sticky Policies

Most of the protocol flows of TAS³ use industry standard Web Services bindings and Web Services payload protocols. It is an explicit design goal that existing services are enabled with minor disruption.

A pertinent problem with existing payload service protocols is how to express the sticky policies that generally have to be bound to the data with a digital signature. Following approaches have been identified

1. Treat all data in one request-response pair as having the same Sticky Policies. In this cases relatively nonintrusive methods like SOAP headers and LDAP controls can be used to indicate the sticky policies. We call this Security Header (SH) approach. This approach is already available as <UsageDirective> SOAP header defined in [IDWSF08].

2. Use the extension points of the payload protocol to express the Sticky Policies. We call this approach Application Protocol Enhancement (APE), see [TAS3D71IdMAnAz] section 8.2. This approach gives granular Sticky Policies that are naturally associated with the data and does not alter top levels of protocol processing. If client and server are updated to understand this scheme then it works well. Eventually new payload protocols should be specified with TAS³ APE feature built in. A danger of this approach is that if the client is not updated, it may just silently ignore the Sticky Policies.

3. Expand the data model to carry sticky policies. This is really a special case of APE with similar merits and problems. One benefit is that it is sometimes easier to extend a datamodel than a protocol.

4. Encapsulating Security Layer (ESL), see [TAS3D71IdMAnAz] section 8.2. Wrap the payload protocol in a TAS³ defined encapsulating protocol that contains all the TAS³ specifics and in particular the sticky policies. Advantages of this approach include
   - The encapsulated protocol does not need to be modified at all
   - Possibility to add sticky policies to protocols that do not offer extension points and that are not under control of the implementer.

Disadvantages of this approach are
   - More invasive on outer layers of the protocol stack. This may make it difficult to integrate to existing protocol stack.
   - If the payload protocol is not SOAP, or otherwise has poor impedance match to the TAS³ ESL protocol, then integration may be impossible.
   - Association of the sticky policies to the data will require awkward correlation of data items to the policies. In particular, if the data does not have item specific IDs, it may be necessary to resort to use of techniques such as [XPATH99].

Given the multiplicity of approaches, each with its merits, the problem arises as to which one should be used. Luckily all can coexist in the same Trust Network. This is made possible by expressing different approaches as different Service Types so that it is possible to discover services that make the approach supported by the client possible. Another approach is to use autodetection, observing the namespaces and elements passed in the SOAP payload to determine which one is being used.

Which of the approaches works best and difficulty of supporting several in parallel are topics of active research in TAS³. A future version of this document will give better guidance for selection of the approach. Currently (May 2009) the APE approach has been successfully used in situations where payload protocol was under our control. The ESL approach has been prototyped, but the specifics of this protocol are still to be determined.
4.2 Legacy Integration Strategy

For TAS³ architecture to be useful, it needs to be adopted. To adopt TAS³ an existing application faces some implementation choices.

Figure 4.1: Application Integration: PEP implemented directly in application.

Conceptually simplest, but in terms of new code to write probably most costly approach is shown in Fig-4.1: just build a TAS³ compliant PEP into the legacy application. This approach has the advantage of allowing full control over the enforcement process, including the inputs to the Master PDP. The disadvantage is the learning curve to learn TAS³ architecture in sufficient detail to implement it correctly and to get certified.

Figure 4.2: Application Integration: ADPEP implemented in application itself.

Fig-4.2 depicts a slightly different strategy where application only implements a simple PEP, which then communicates with the Application Indepenent PEP (AIPEP) supplied by the TAS³ Project. In this approach, the AIPEP component handles most of the TAS³ specific parts and can be an already certified component, making compliance certification easier.
However, in this model the need to communicate between AIPEP and ADPEP arises. The communication pattern could be either Encapsulating Security Layer (ESL) or Application Protocol Enhancement (APE), as described in the Section ??

Fig-4.3 illustrates some specific integration strategies with the intent of enabling legacy data sources that can not be modified. In (A) the SOA Gateway evolves to support TAS3 architecture, in (B) the SOA GW is frontended by the WP9 database which supports the TAS3 architecture aspects. If export of the legacy data is an option, then it may be simplest to import the data to WP9 database and dispense with the legacy data source entirely (C).

![Diagram of Service Responder, Stack, AIPEP, and Application Dependent PEP](image)

**Figure 4.3:** Application Integration using ADPEP and (A) SOA Gateway, (B) WP9 DB as frontend to SOA GW, (C) WP9 database.

### 4.3 ADPEP

The Application Dependent Policy Enforcement Point (ADPEP) is a gateway that provides access to the TAS3 infrastructure for applications like web applications with a web frontend, business process engines, databases or repositories and many other systems, which are either requesting or responding over a TAS3 secured and trusted channel. Per section 2.2.1 (see also Fig-2.2), the ADPEP belongs to the Front End Services and Web Service components inside the Payload boundary.

As described in Section 2.2.2, the ADPEP is divided into two different types of Application Dependent Policy Enforcement Points:

1. ServiceRequester ADPEP: This web service is part of the Front End Services. Internally, the ServiceRequester ADPEP constitutes together with the Stack, the Service Requester. The Stack handles SOAP protocol details. The Application Independent Policy Enforcement Point (AIPEP) contacts Master PDP, which contacts different PDPs like User PDP, Organization PDP or a Trust PDP to decide whether a requested is trusted or not.

The main task of the ServiceRequester ADPEP is to collect all required information for an appropriate request that has to be checked by the TAS3 authorization infrastructure. Further information about the payload, which builds up the request, can be found in [TAS3D81RepoSW] figure 8. Common information about the functionalities of the ServiceRequester ADPEP can be found in [TAS3D81RepoSW] and in [TAS3D83CliSW].

The next steps before sending the request are done by the 'Stack’. As mentioned before, the 'Stack’ (and its main component: the AIPEP) is application independent. Its main task is the preparation of the request. The message has to be signed and augmented according to web services binding. WP4,
WP5 and especially WP7 work on this security related part of the service requester. Whereas WP8 is responsible for the application dependent part.

2. ServiceResponder ADPEP: This second application dependent service, which functions as responder, is part of the Service Responder component in the Web Service boundary (see Fig-2.3). In analogy to the ServiceRequester ADPEP, the ServiceResponder ADPEP also needs the 'Stack' (with AIPEP and its underlying PDPs) to function correctly. That means, signing and preparation of the message according to the web service binding, the policy checks and the communication with the 'Trust policy decision point', as done by the 'Stack components'.

The main task of the ServiceResponder ADPEP is to receive requests, route them in an appropriate way to the 'Service Application' and then send back the response to the requester. More details about the functionalities of the ServiceResponder ADPEP can be found in [TAS3D81Rep] in chapter 3.2.

Auxiliary components in the both ADPEP Services

To fulfil the mentioned functions of the both ADPEP Services (Requester and Responder), some auxiliary services are required. These services belong to tasks (Task 8.3 - see DoW), which are documented in [TAS3D82BackOffice]. These services neither store person related data nor serve the user directly. They provide ontologies and metadata, perform search and aggregation operations and transform data into specific formats. The back office services are a component of the TAS3 Trusted Application Infrastructure but not of the core TAS3 Trust and Security Infrastructure.

The main Auxiliary or Back Office Services and Components are:

- The Generic Data Format ([TAS3D82BackOffice], section 2.1.2) used to store data in TAS3 repositories
- Services to transform ([TAS3D82BackOffice], section 2.1) data from a custom source format to the Generic Data Format and from the Generic Data Format to a format, which is requested (and supported).
- Aggregation Service ([TAS3D82BackOffice], section 2.2) and Policy Aggregation ([TAS3D82BackOffice], chapter 8)
- Request Logger Service ([TAS3D82BackOffice], section 3.2) to store information on requests issued and responses received by TAS3 web services for auditing and maintenance purposes

4.4 Reputation Feedback

The workflows have feedback collection step. This feedback is collected by a user interface and then sent to Trust and Reputation Server by means of a web service call (e.g. SOAP). The call will contain, in addition to the feedback per se, workflow context information, such as the role in which the user acts in as well as pseudonym of the user at the Trust and Reputation Service (see below). If user is giving feedback about another user, he will know the other user's pseudonym through the business process context. This architecture allows the Trust and Reputation Server to correlate feedback from different sources without the sources having a correlation handle for the user.

The pseudonym of the user at the Trust and Reputation Server is obtained by means of normal discovery and ID mapping (see chained flow).

---

1Marc: not applicable for eHealth because of legal issues.
4.5 Business Process Registration

When user is about to start participating in a business process, he can go to the Dashboard and request to view a declaration about the data needs and other participants of the business process. Such declaration is prepared from the Business Process Model.

User can then edit his policy to allow the data needs to be satisfied and shared with the other participants of the process.

Each business process model can be viewed as a high level service and can be described by a service type. This makes it possible to discover Service Providers who implement the business process.
5 Using Business Process Modelling to Configure the Components

This section addresses Reqs. D1.2-3.2-ModelDrivenCfg, D1.2-3.12-SPManifest, D1.2-6.3-WhatHowWhyWho, and D1.2-6.4-Min.

The TAS³ architecture covers a lot of functionality and some of this functionality needs to be configured carefully to match each other to ensure smooth operation from the perspective of the users, such smooth operation is perceived by users as dependability and trustworthiness, so it is a prerequisite for good public image of a Trust Network.

Correct configuration will also be essential for ensuring that services function securely. Given that most security technology is quite brittle and even minute misconfigurations lead to failure, there will be operational and commercial pressure to turn off those nonfunctional, but essential features that appear to be "causing trouble". This is an extremely dangerous slippery slope that any Trust Network MUST avoid. Liberty Alliance and SAML Interoperability and Certification programmes have clearly demonstrated this to be a real peril. Therefore it is necessary that it is possible to correctly configure the trust network such that it will work right on the first try.

Complexity of a typical Trust Network, along with all of its member systems, is of such a high degree that it is infeasible to configure it sufficiently correctly by a manual approach. Humans make mistakes. An automated, model driven configuration is the only way to create accurate and correct configuration.

The corner stone is Business Process Modelling. From this model, which exists both at the top Trust Network level and at the organizational level, it should be possible to derive the following outputs:

1. Circle of Trust parameters to facilitate federation and SSO configuration
   a. White list of roots of trust for both authentication and authorization
   b. Trusted Certificates for TLS [RFC3548] and Signing
   c. Metadata for entities

2. Declarative Statements about attribute needs of the Clients as well as policies under which providers are willing to release attributes. This output will come in the form of [CARML] and [AAPML] files, see further [IGF], that can be used to automatically configure IGF enabled layers of Client Request PEP and Provider Request PEP.

3. Some of the top level policies that apply to the Trust Network and its members. This should facilitate configuration of the PDPs.

4. Policies, Business Process Models, and Interface Descriptions (e.g. WSDL) that are needed as input for the Automated Compliance Validation.

5. Business Process Models that are needed as input for Business Process Visualization at the Dashboard.

6. Policy and business process model descriptions needed by the Configuration and IT infrastructure management, e.g. CMDB, ITIL, etc.

Contractual information behind a business process will influence the business process model itself and has an impact on the security rules, roles, and policies related to the business process.

Security rules that guide selection of web services and use of secure entities (data), i.e. influence the discovery service.

Security exceptions during business processes will raise an exception. Unhandled exceptions will block or break the process (go to an operator or help desk). The challenge is to handle the exceptions by explicit routines in the business process modelled routines or in some cases by using alternative paths (i.e.
subprocesses) to allow the process to complete or even to look ahead and avoid exceptions before they occur by such means.

Business processes can have more complex topology than just trees.

There is no centralized log, just references to local logs are passed out.
6 Oversight and Monitoring

For a TAS³ compliant Trust Network to gain a trustworthy reputation and to ensure that belonging to the Trust Network really enables lower cost of operation through lesser fraud, improved trust, and ultimately less need for formal audits, it must take proactive and mandatory activities to monitor its activities and stop any fraudulent practises before they become a problem, ideally even before they become publicly known.

This section addresses Reqs. D1.2-2.11-Transp, D1.2-2.12-Compr, D1.2-2.15-Resp, D1.2-2.16-Mitigate, D1.2-2.17-AuditUntamp, D1.2-2.21-DataProtLaw, D1.2-2.22-GovtAccess, D1.2-12.13-Vfy, and D1.2-12.15-Valid.

In TAS³, the monitoring should happen at levels of

1. Continued automated, robotic, testing that compares results to both modelled expectations and past results. This is one of the focus areas of TAS³. See: On-line Compliance Testing (OCT).

2. Operations monitoring to determine upness and performance of services, as well as detection of anomalies. Trouble ticket system for reporting and rectification of operational errors, as well as intrusion detection scans and monitoring are included here as well. Use of industry standard solutions is recommended as TAS³ does not plan additional research in this area.

3. Log audit. Some part of log audit is handled in operations monitoring, above, but logs will contain a wealth of additional information, such as usage patterns to inform new investment and areas of innovation, which can be extracted using data mining techniques. Use of industry standard solutions is encouraged in general as the only connection with TAS³ research is in the area of gathering inputs for reputation scoring.

4. Formal compliance audits should occasionally be carried out manually to ensure that the automated monitoring and audit mechanisms, above, are functioning correctly. These audits may be mandated by legislation or by governance agreement and are typically fairly costly affairs with reputable outside consultants specializing in organizational and IT audits. TAS³ contribution for this area stems from recommendations and guidelines of the project legal team.

5. Administrative Oversight. The Trust Guarantor will take necessary administrative steps to ensure that the Trust Network is adequately monitored, mostly automatically, but with necessary and timely manual intervention. The Trust Guarantor may, according to the Governance Agreement, be monitored by an Advisory Board, Management Board, and ultimately General Assembly.

Section of 4.3.7 "Management" of [NexofRA09] discusses the need for management interfaces in services components. TAS³ is compatible with these requirements.

6.1 Dashboard


6.2 Right of Access, Rectification, and Deletion

Data subject has right to know what data is held about himself and often may rectify or delete it (however, in case of some data such as medical data, the subject may not be allowed to edit or delete the data).
In online transactions it is important to be able to tell where it originated so that the user can contact the right authority.

### 6.2.1 Identification of Originating Authority

Primary means of addressing Right of Access is mandatory identification of the authority from where the data originated. TAS³ attribute authorities MUST identify themselves in the data set. One of the following approaches are acceptable:

1. If data is conveyed in SAML assertion and the origin of the data is the same as the assertion’s Issuer, then the assertion’s Issuer field is taken as sufficient identification of the authority.

2. If data is conveyed in X509 attribute certificate and the origin of the data is the same as the certificate’s Issuer, then the certificate’s Issuer field is taken as sufficient identification of the authority.

3. Include in the data set attribute named `urn:tas3:issuer` whose value is the Entity ID of the issuer.

The Right of Access, Rectification, and Deletion ultimately needs to be satisfied at the origin of the data. To facilitate this process, the Service Providers that are consumers or users of the data MUST display the identity of the origin of any given data item or data set. Another way for the user to find out the origin of the data used in transactions is to see it in the Dashboard.

Either way, the user is then expected to direct his Right of Access, Rectification, or Deletion requests directly to the origin. User MAY request deletion of the local copy from the SP using the data, but the using SP is not responsible for correcting the data at the origin. Instead the user really needs to contact the origin.

### 6.2.2 Facilitating Self Service Interface to Right of Access

To facilitate self service interfaces for Right of Access, Rectification, or Deletion, the data set SHOULD include the attribute `urn:tas3:issuer:selfmgmt` whose value is a URL to the user self management web GUI, where user MUST be able to satisfy Right of Access, and MAY be able to satisfy Rectification and Deletion, subject to authorization policies applicable (e.g. user may not be allowed to edit his health records, only his doctor can).

### 6.2.3 Propagation of Rectifications by the Originating Authority

The originating authority MUST keep record of the parties to which it released data. The concerned user SHOULD be able to query this record using the self service interface. The records need to be kept in the minimum for the duration specified by trust network, but in no case for less than 6 hours.

When a data user SP requests data, it SHOULD also create a subscription to receive rectifications to the data. The Originating Authority’s records MUST show if the requester created subscription, i.e. whether it is possible to propagate rectifications to it. The Originating Authority MAY refuse data requests, or attach short data retention obligations to the requests, from SPs that do not create subscription to receive rectifications.

When user rectifies or deletes data the authority MUST check its records for SPs that have received the data and by virtue of emitted data retention obligations MAY still have copies of the data. If such party has subscribed to the changes, the authority MUST propagate the data, unless user has given explicit instructions to the contrary.

Soliciting user’s instructions on this matter is OPTIONAL. If the receiving party has not subscribed for the updates, and the user has not instructed to withhold propagation, the authority MUST log to audit bus a notice of inability to propagate change. The Dashboard will pick up these notices and allow the user to see them so that the user is aware of incomplete propagation.

When propagating rectifications, the receiving party MUST further propagate, if it has given data to any further parties.
6.3 On-line Compliance Testing

This section addresses Reqs. D1.2-6.14-Compat, D1.2-6.15-MinPolicy, and D1.2-12.16-OnlineTst.

Implementation of SOA based applications result from the integration of several services. Services composing an application can change at run-time without informing all the other services integrated in the application. Furthermore, features like dynamic binding, or context-dependency prevent knowing before run-time the actual interaction among service instances.

Speaking in general terms, services are typically controlled and owned by different organizations. Thus, dealing with architectures that are not under the full control of one organization, means that the service lifecycle cannot be structured in well-defined development stages. In particular, for a (composite) service it is not clear when testing activities starts or should end.

To ensure trust and dependability the TAS³ architecture must also include adequate technology and actors to test that services within a TAS³ Trust Network behave in compliance with their expected specifications. Such testing activities must be performed on-line by special TAS³ guards, verifying that the services with a choreography actually behave as expected.

To achieve trustworthy SOA, there is the need to develop and use a methodology and tools supporting the "perpetual" (i.e. event-driven, periodically) and automatic testing of software services. The benefits with the "perpetual" and automatic testing we envision:

- repeatability of testing (improving the efficiency and the efficacy of the test)
- increase the quality and the trust perceived by the users of the service

![Figure 6.1: Overview of On-line Compliance Testing](image)

The extent to which compliance is tested may vary, also depending on the registration information that should accompany services (e.g. models describing interfaces, policies, usage, etc.), which is part of the governance contract.

A minimal assumption is that services should access and perform within a TAS³ infrastructure according to an explicitly declared set of policies and that the infrastructure should not allow violation to the declared policy, or at least should recognize such violation. Testing is applied in order to reduce the risk that services within a TAS³ infrastructure will get in contacts with unreliable services. Therefore services
within a TAS³ compliant infrastructure will be regularly submitted to testing session aiming to assess that a service does not break its policy.

As important remark, we advise that this on-line testing approach does not prevent the execution of canonical off-line testing activities (e.g. where the service is tested by its developer trying to anticipate possible usage scenarios), rather it is an additional mean to increase the trustworthiness of the TAS³ architecture.

### 6.3.1 Involved Actors

On-line Compliance Testing impacts on the following of actors of the TAS³ scenario.

- **Ecosystem**
  - Service Provider
  - Reputation Providers
  - Software Certification agencies
  - Deployment certification and audit agencies
  - Compliance Authority

- **Components**
  - Web Service (WSP)
  - IdP (Identity Provider)
  - Service Registry
  - Id Mapper
  - Linking Service
  - Organizational PDPs
  - Trust Network PDP

### 6.3.2 On-line Testing Process and Architecture

The TAS³ architecture includes an on-line testing infrastructure, in which, the TAS3 services are tested according to a set of published models describing specific behavioural characteristics of the service itself. In the following, the term OCT refers both to the on-line compliance testing process and to the infrastructure that implements it.

With respect to the current scope of the TAS³ architecture, the compliance testing functionality requires that each service exposes within the TAS³ choreography its public interface (i.e. its exported operations) and the public policies it will comply with (or a references to them). In particular, each service is tested on-line when it requests to be registered to a TAS³ directory service. Further on-line test sessions can be activated by the compliance testing (i.e. Trust Network level) directory service either in event-driven or periodic fashion.

Directory Services within a TAS³ architecture interact with testing components to detect services failures. The compliance testing increases trust both on the TAS³ architecture and the linked services. A software service willing to be registered to a TAS³ directory service, may also have to comply with other policies that are not publicly manifested. In this case the service will not be tested with respect to such policies since such behaviour is hidden from the external interfaces of the service.

During the on-line compliance testing process, references to the service should not be retrievable by means of the directory service. At the end of the compliance verification process, if and only if all the tests have been successfully executed, service references will be listed by the directory service.

During the on-line testing, the OCT activates tester robots. Each tester invokes the service under test simulating service requests with identity credentials taken from a pre-packed identity test suite. The tester
robot collects the service reply (i.e. either a response message, or a deny access), and compares it with the expected results: a difference between the service reply and the expected result reveals a mismatch between how the service policy is manifested and how the policy is implemented within the service.

Note that the TAS³ architecture has a precise requirement that error messages returned after a request for a resource (e.g. ”access denied” message) must be identifiable as such. Applications might masquerade error messages for user-friendliness (e.g. they could produce a ”pretty formatted” page); nonetheless, the TAS³ architecture needs to be able to unambiguously recognize error messages without the need to delve into the semantics of the payload of the message. This is accomplished by each application declaring to the on-line compliance testing infrastructure at least one successful and one failing test case with exact description of what the messages look like and what are the relevant parts.

In real life scenarios, a service under test may need to access external services when invoked by the tester robot. Indeed, in some cases a testing interaction between the service under test and externally invoked service may have permanent effects (e.g. on a stateful resource). Let’s consider that the service under test queries the directory service to lookup a relevant end point. In this case, OCT should consider that the registry may return a reference to a Proxy version of the required service: this service will implement the same interface as the required service. Doing so, the real implementation of registered services is hidden to those services waiting for compliance validation - a useful feature while project is ongoing and full service is yet unavailable.

In such cases, the directory service and OCT have to be able to link to an existing service proxy or to generate new ones. Obviously this will increase the complexity of the framework and asks for the provisioning of service description models suitable for automatic generation of service stubs.

Fig-6.2 depicts a UML Diagram describing the components of the On-line Compliance Testing framework. In the following we list a detailed descriptions of each component:

- Compliance Validator Discovery Service: according to the architecture given in Fig-??, this component enhances the functionality provided by the Discovery Service. In particular, the Compliance Validator Discovery Service is a Discovery Service able to apply the compliance validation at runtime. The Compliance Validator Discovery Service aggregates three sub-components: the Compliance Validator, the Proxy Factory and the Pending Services DB.

- Compliance Validator: this component activates the testing session when a service makes a request for being included in a TAS³ infrastructure. The testing session will result in a sequence of invocations to the services requesting to be registered. In case the testing session does not highlight any error the service will be registered otherwise the request will be rejected.

- Tester Robot: this is the component that actually runs the test on a given service. In particular, the Compliance Validator activates the Tester Robot passing to it a reference to the service that needs to be tested and to the corresponding test suites to use.

- Request Generator: this component defines which is the next invocation to make to the service under test in order to assess its correctness;

- Test Checker: this is the component that checks that the replies of the service under test actually conform to what is specified;

- Test Attribute Generator: this component defines which attribute values are to be used in the testing invocations.

- Front Channel Tester: this component extends the Tester Robot adding to the tester component specific features to interact with the front channel of a service application in TAS³. Since this component is a Tester Robot, it has all the features that a Tester Robot has, including the relationship with the other components (e.g. Request Generator, Test Checker, Test Attribute Generator).

- Back Channel Tester: this component extends the Tester Robot adding to the tester component specific features to interact with the back channel of a service application in TAS³. Since this
For example according to the SP-Initiated SSO with Redirect and POST Bindings schema at Fig.12 of the SAML Specification, Likely SAML Tokens can be extracted from the directory service during test preparation. Extracted from the IdP during test preparation.

Figure 6.2: UML Component Diagram of the On-line Compliance Testing (OCT).

Component is a Tester Robot, it has all the features that a Tester Robot has, including the relationship...
with the other components (e.g. Request Generator, Test Checker, Test Attribute Generator).

- DB of Roles as Signed Test Response: this DB contains the identity test suite that the tester can use to test other services simulating a different identity.

- DB Test Report: in this DB the compliance validator logs the result of a testing session and the possible failures highlighted.

- Proxy Factory: this component automatically generates proxy services to simulate already registered services.

- Pending Services DB: this DB will contain the identities of services that requested to enter a TAS$^3$ infrastructure precinct but still did not pass the testing session. Services in such DB are not returned as result of a discovery request. Identities are removed from this DB when the corresponding testing session is terminated.

Note that, each entity (i.e. components or artifacts) in the diagram, has a role with respect to the domain organization. Specifically, according to the general architecture depicted in Fig-??:

- The artifacts Public Interface, and Public Policy, are entities within the Modelling & Configuration Management area,

- The OCT components (Compliance Validator Discovery Service, Compliance Validator, Tester Robot, Request Generator, Test Checker, Test Attribute Generator, Front Channel Tester, Back Channel Tester, DB of Roles, DB Test Report, Proxy Factory, and Pending Services DB) are entities within the Audit & Monitor area,

- The components IDP, Discovery Service, and Web Service, are entities within the Runtime & Enforcement area.
6.4 Operations Monitoring and Intrusion Detection

[NexofRA09], section 4.3.7 "Management", paragraph 4, highlights the need for operational monitoring. While such monitoring is not a requirement for technical interoperability of TAS³ framework, it will be necessary to maintain reputation of TAS³ Service Provider and/or Trust Guarantor. This topic, which addresses Req. D1.2-1.6, is not an area for TAS³ research work. Consequently:

1. Standard operations monitoring approaches such as SNMP [RFC1157] and Nagios [Nagios] SHOULD be implemented.

2. Each organization in the Trust Network MUST be protected by network level firewall or packet filter. Any deny events from the firewall SHOULD be fed to the Intrusion Detection Channel of the Audit Event Bus.

3. Each organization in the Trust Network SHOULD operate an Intrusion Detection System (IDS) to
   a. Detect well known attacks (e.g. ping of death)
   b. Port scanning
   c. Abusive patterns of usage

Any suspicious events from the IDS SHOULD be fed to the Intrusion Detection Channel of the Audit Event Bus.

6.5 Log Audit

This section addresses Reqs. D1.2-2.17-AuditUntamp, D1.2-3.3-Dash, D1.2-6.10-Redress, and D1.2-7.21-Safe.

Log audit has several goals

1. In case of attempted repudiation, prove that events happened

2. For investigation, browse and visualize events so that a human investigator can get a relevant and sufficient overview.

3. On an ongoing basis automatically detect noncompliant events

4. On an ongoing basis ensure that the systems are functioning correctly

5. Provide statistics about users and their behaviour

6. Provide statistics about system use and behaviour

7. Provide baseline of information that allows trust, security, and access control mechanisms to be cross checked

Log audit raises several issues

A. Collection

B. Distribution

C. Privacy

D. Retention

E. Falsification
F. Omission

G. Denial of Service and overwhelming the logging system

The log audit could also be used for billing in some circumstances, but in general we recommend that billing systems be built separately so that they can be cross checked against the logging to detect errors.

A taxonomy of audit events is presented in Annex A "Enumeration of Audit Events". Some design requirements for Audit Log Files are discussed in [TAS3D42Repo]. Further requirements include

I. Append mode of access: Only append mode of access should be allowed, so that users or applications cannot rewind an audit log file and delete or modify information that has already been stored there.

II. Authorised writing: Only authorised parties should be able to append log records to the audit trail. Though unauthorised applications or attackers may gain access to the audit trail and try to append fake log records to the audit trail, or modify or remove the audit trail, this should be detected by the tamper detection mechanism.

III. Timestamps: Every record in the audit trail should be timestamped to provide a trusted record of when the audit data was received. We note that if the audit service is trusted to record the audit data without tampering with it, then it should also be trusted to append the correct time to the data. Therefore we do not require a secure time stamping service.

IV. Secure communication: if the audit service operates as a web service then there should be secure communications between the clients and the server in order to ensure tamper resistance, data integrity and authorised connection.

V. Secure storage on untrusted media: Since an audit trail may be viewed on untrusted machines, the security mechanisms should ensure persistent and resilient storage of the audit trail, and ensure detection of tampering of the audit trail - modification, deletion, insertion, truncation, or replacement. If tampering is detected, the audit service should be able to notify the security auditor.

VI. Support multiple simultaneous clients: The audit service should be easily and conveniently accessible and it should be able to serve multiple client applications simultaneously.

VII. Logging efficiency: The computational work and the storage size required by the audit service should be as efficient as possible.

VIII. Contents transparency: the audit service should be able to record any digital content coming from any repository service.

IX. Authorised reading: Since the audit trail may contain personal or sensitive information, then the audit service should ensure that only authorised applications or people have the privilege to read the audit trail. The audit trail may be encrypted to further protect confidentiality.

6.5.1 Log Collection and Storage

This section addresses Req. D1.2-2.22-GovtAccess.

In the TAS3 architecture the audit trail is collected and stored locally primarily at the system entities, such as SPs, IdPs, the IM, and the like or near them in the organizations that operate these entities. Everyone that collects a log is bound by a Governance Agreement so that responsible behaviour can be enforced when technical solutions fall short in some area of protection.

The log events originate in various components at various times, see Annex A "Enumeration of Audit Events" for an idea of the types of events that will be generated. For example, Web Services Stack component will check signatures on the tokens (assertions) that are presented and log both positive and negative outcomes.
The system entities that collect the audit trail or the centralized audit function of the organization report the events in *summary form*, essentially just *pointers* to the actual audit records, to the Audit Event Bus. Each component may keep its local log in its own format (in future we may provide standard format), but the summary logging to the Audit Event Bus will follow TAS³ standard format (this format will be presented in a future version of this architecture). To facilitate standard format summary logging, TAS³ may provide a reusable software library.

The Audit Event Bus is divided in channels to which different events are broadcast. This allows *minimal exposure* as subscriptions can be on the basis of only relevant events. The subscriptions can also be controlled such that only authorized parties with "need to know" can see certain types of events (see req IX above).

The Audit Event Bus is potentially implemented as part of a more generic Event Bus infrastructure, but due to special privacy and security requirements, Audit Events MUST NOT be mixed with other business messages, unless in encrypted form. If the generic event bus supports an encrypted private channel, a VPN if you like, then sharing of the infrastructure may be possible.

The Audit Bus infrastructure MUST be free of conflicts of interest. In particular, it should not be operated by one of the SPs. In case the Event Bus sharing is implemented, then the operator of the shared infrastructure MUST be free of conflict of interest as well.

### 6.5.2 Privacy Issues: What to Collect and What to Report

This section to be fleshed out in project month M30 release of D2.1. It will satisfy Req. *D1.2-4.2-BPPrivacy*.

The main issues are

1. Avoid logging anything that could become a correlation handle
2. Avoid logging PII unless absolutely necessary

Generally a lot of detail will be logged locally. This will include the tokens used in identification the user, usually in pseudonymous form as well as the PII handled by the Service Provider. This detail tends to be necessary to legally protect the Service Provider.

### 6.6 Formal Compliance Audits

This section will be developed in the D2.1 of M30.

- Legal compliance
- Penetration testing
- Disaster recovery exercises
- Completeness of the Audit Trail

See req 6.65.

### 6.7 Administrative Oversight

This section partially addresses Req. *D1.2-6.10-Redress*.

Administrative oversight and stake holder issues are covered in [TAS3BIZ].

This section will be developed in the D2.1 of M30.
7 Conclusion: TAS$^3$ is Secure and Trustworthy

Comprehensive approach of the TAS$^3$ architecture and framework achieves real and tangible overall security and trustworthiness gains when compared with state of the art for multiplayer networks of comparable size. TAS$^3$ features that contribute to this are

1. Legal concerns are built-in from the ground up
2. A comprehensive and strong digitally signed audit trail
3. A conditionally pseudonymous audit trail to guarantee the privacy of Users who play by the rules, while allowing abuse to be exposed through collaboration of Service Providers.
4. A fully pseudonymous design at all layers to protect user privacy
5. Fully encrypted and digitally signed messages using strong algorithms
6. Based on state-of-the-art Single Sign-On protocol standard (SAML 2.0) which has had extensive security review
   - Extensive security review and scrutiny already done
   - Multiple commercial and open source implementations that are mature.
   - Certification program for implementations further ensures quality
7. Based on state-of-the-art Identity Web Service Protocol standards (ID-WSF 2.0) which have had extensive security review
   - Extensive security review and scrutiny already done
   - Multiple commercial and open source implementations
   - Certification program for implementations further ensures quality
8. Enhanced authorization infrastructure which significantly improves upon the current XACMLv2 standard
   - Extensive security review and scrutiny already done
   - Multiple commercial and open source implementations
9. Ability to use risk control and reputation
10. Use of ontologies to ensure consistent interpretation of data and authorization rules
11. On-line Compliance Testing for early detection of discrepancies and problems
12. Business Process Modelling driven configuration to ensure consistently correct configuration
13. TAS$^3$ has performed a systematic threat analysis (see Annex F) to ensure that the architecture addresses the widest possible range of security and privacy threats.
14. Software engineering techniques used by the project to consistently achieve high quality and absence of security bugs in the software components that are TAS$^3$ deliverables.

TAS$^3$ Architecture is novel as a blueprint that brings together identity management, attribute based access control, business process modelling, and dynamic trust. The architecture, with Annex A, acts as an interoperability profile for various standards based protocols covering these areas. Other areas of innovation are user transparency features like Dashboard, user accessible audit trail, and automated compliance validation; privacy protection using sticky policies; marriage of trust and privacy Negotiation with discovery and trust scoring; secure dynamic business processes; and built-in first class support for delegation.
Annex A: Enumeration of Audit Events

To understand the wealth of audit trail data we start by enumerating them all:

1. Session Events Channel:
   a. Session creation (possibly even an anonymous session)
   b. Session upgrade (e.g. SSO on an anonymous session, step-up auth)
   c. Session refresh
   d. Session termination
   e. Session expiry
   f. Session revival (if appropriate, could be used as a factor in authentication)

2. User Authentication Events Channel:
   a. Positive
   b. Failure with Retry
   c. Definitive Failure

3. Token Issuing Channel:
   a. Tokens issued with:
      i. Issuer
      ii. Subject
      iii. Audience
      iv. Policy constraints
      v. Validity time and/or usage count
      vi. General content of the token
   b. Token validation at relying party
   c. Token use, to the appropriate extent
   d. Token revocation when applicable

4. Authorization Channel:
   a. Az request parameters
   b. Az decision returned
   c. Obligations
   d. Promises to respect obligations

5. Service Requester Channel:
   a. Choice of Service Provider
      i. Discovery
      ii. Hardwired choice of Service
      iii. Automated or algorithmic Choice of Service
      iv. Choice of Service solicited from the User
   b. Trust negotiation steps
   c. Consent to send data, consent points, how was the answer obtained (e.g. automatic vs. interaction)
d. Service Call event
   i. Signature preparation, including choice of signing key
   ii. Log of content of the message
   iii. Peer authentication
   iv. Success or failure to send message

e. Service Call exception
   i. Redirect or end point change
   ii. Recredentialing
   iii. Interaction requested
   iv. Replay after interaction
   v. Dry-run

f. Service Call Response
   i. Log of content of the message
   ii. Peer authentication (usually by Request-Response pattern)
   iii. Success or failure to receive message

g. Service Call Response exception
   i. Failures, as detailed on the Faults Channel
   ii. Application layer success or failure

h. Obligations processing
   i. Presence of obligation
   ii. Specific processing steps
   iii. Failure to process obligation

6. Service Responder Channel:
   a. Trust establishment and trust negotiation steps
   b. Request Acceptance
   c. Response filtering and authorization decision
   d. Attachment of obligations

7. PII Collection Channel

8. PII Release Channel

9. User Registration Channel:
   a. Register
   b. Modify
   c. Deregister

10. SP Registration Channel:
   a. Register
   b. Modify
   c. Change of Control
   d. Deregister

11. User Reputation Channel:
12. Service Reputation Channel:
   a. Explicit complaint or praise
   b. Other events that affect reputation

13. Browsing Event Channel (usually not shared)

14. Faults Channel:
   a. Malformed protocol message
   b. Insufficient sec mech
   c. Signature verification fault
      i. Malformed
      ii. Crypto (public key or hash)
      iii. Certificate validity (missing CA trust chain)
   d. Inappropriate use
      i. Audience
      ii. Constraints
   e. Expired tokens
   f. Replay of message or token
   g. Unsolicited message
   h. Missing database entry
      i. Explicit fault report

15. DoS Channel:
   a. Invocation frequency alert
   b. Data volume alert
   c. Explicit DoS report (e.g. from monitoring organizations)

16. Intrusion Detection System and Firewall ACL Channel:
   a. Scan alert
   b. Attack fingerprint alert
   c. Firewall deny rule triggered

17. Operations monitoring Channel:
   a. Server / Service
      i. Up
      ii. Down
      iii. Scheduled downtime
      iv. Congested
      v. Retry
      vi. Fail Over
18. Audit Operation Channel (very restricted circulation):
   a. Undertaking audits
   b. Outcomes of audit


20. Customer Care Event Channel
Annex B: TAS³ Risk Assessment

B.1 Executive Summary

Threat modeling aims at studying the potential problems and attacks against a system in order to document how attacks are mitigated. In this document, we will analyze threats and suggest solutions in order to produce internal threat analysis document. It provides an assessment on how threats are mitigated and which threats are remaining. It only provides the threat model of components developed in this research project. TAS³ is an integration project with the vision of implementing an architecture. Such an architecture-driven approach means to use a methodology where the assessment can be done by component.

The document contains a brief overview of some existing methods, the description of the methodology adopted (derived from the NIST800-30 Microsoft methodology) and the risk assessment by components. In the deliverable D2.1, the TAS³ Architecture has already presented some inputs in this direction (Annex B that summarizes the threats that the TAS³ architecture is designed to protect against).

B.2 Introduction

This section provides a brief overview over some of the existing methods and standards related to risk analysis. It concludes by identifying some issues that to little degree are covered by the current state of the art. Most of the existing methods have been presented in [?].

B.3 Risk Analysis Methods

B.3.1 OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)

OCTAVE [Alberts01] is a general approach for evaluating and managing information security risks. As information security includes issues related to both business and technology, an inter-disciplinary analysis team that includes people from both the business units and the IT department performs the evaluation. The evaluation is performed in three phases:

Phase 1 Build Asset-Based Threat Profiles. The analysis team identifies what is important to the organization, i.e. what are the information-related assets of the organization, and reviews what is currently being done to protect those assets. The analysis team also selects the critical assets that are most important to the organization. The team then describes security requirements for the critical assets and identifies threats to the critical assets.

Phase 2 Identify Infrastructure Vulnerabilities. The team evaluates the information infrastructure and identifies key information technology systems and components related to each critical asset. Key components are examined for weaknesses (technology vulnerabilities) that can lead to unauthorized action against critical assets.

Phase 3 Develop Security Strategy and Plans. The analysis team identifies risks to the organization’s critical assets and decides what to do about them. The team creates a protection strategy for the organization and mitigation plans to address the risks to the critical assets, based upon an analysis of the information gathered.

OCTAVE provides a snapshot analysis at a given point in time. Therefore, OCTAVE advises that the organization either performs new evaluations periodically or triggered by major events, such as corporate reorganization or redesign of the computing infrastructure. OCTAVE comes with predefined templates for documenting information during the analysis.
B.3.2 CRAMM

CRAMM (CCTA Risk Analysis and Management Method [Siemens10]) provides a stepwise and disciplined risk analysis method that takes both technical and non-technical (e.g. physical and human) aspects of security into account. In its original form, it was adopted as a standard by the U.K. government organization CCTA (Central Computer and Telecommunications Agency). Similarly to OCTAVE, CRAMM defines three stages of analysis:

Stage 1 Asset identification and valuation. The reviewer identifies the physical (e.g. IT hardware), software (e.g. application packages), data (e.g. the information held on the IT system) and location assets that make up the information system. Value is assigned to assets.

Stage 2 Threat and vulnerability assessment. The likelihood of potential problems are identified, taking into consideration both accidental and deliberate threats, such as hacking, viruses, equipment or software failure, wilful damage or terrorism, and errors made by people. Based on this, the risk level is calculated.

Stage 3 Countermeasure selection and recommendation. The measure of risk is evaluated in order to decide what countermeasures should be implemented. CRAMM provides a countermeasure library where a threshold level is associated with the countermeasures, thereby providing aid in the decision whether to implement a given countermeasure.

During a CRAMM analysis, the reviewer gathers information by interviewing the asset owners, system users, technical support staff and security manager. A standardized CRAMM format is used for documenting results, mostly in the form of specialized tables.

B.3.3 Microsoft’s Security Risk Management

Microsoft has developed their own risk guideline [Microsoft06]. This guideline defines the Microsoft Security Risk Management Process, which has four primary phases:

Phase 1 Assessing risk. Data are gathered in order to identify risks to the business, and to prioritize between the risks.

Phase 2 Conducting decision support. Functional requirements to mitigate risks are defined. Control solutions (treatments) are identified and evaluated, and a cost-benefit analysis is performed.

Phase 3 Implementing controls. The chosen control solutions are implemented and deployed.

Phase 4 Measuring program effectiveness. The risk management process is analyzed for effectiveness, and an evaluation of whether the controls provide the expected degree of protection is performed.

As risk management is viewed as an ongoing process, these phases constitutes the parts of a risk management cycle. The guideline also goes further than defining this cycle. For example, it provides lists of common assets, threats and vulnerabilities. However, these are not intended to be comprehensive, and analysts are encouraged to add or delete items as necessary.

For example, the risk management guide for IT systems (NIST800-30) [NIST-SP800-30]. defines a methodology to assess risks while developing an application.

B.3.4 CORAS

CORAS [BraberEA07] is a method for conducting security risk analysis. CORAS provides a customized graphical language for threat and risk modelling, and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis. In this respect CORAS is model-based. Special CORAS diagrams are used for
documenting intermediate results, and for presenting the overall conclusions. The CORAS language is
supported by a structured semantics translating CORAS diagrams into natural language (English) sen-
tences [DahlEA07]. Following the CORAS method, a security risk analysis is conducted in seven steps:

**Step 1** The first step involves an introductory meeting. The main item on the agenda for this meeting is to
get the representatives of the client to present their overall goals of the analysis and the target they
wish to have analyzed. Hence, during the initial step the analysts will gather information based on
the client’s presentations and discussions.

**Step 2** The second step also involves a separate meeting with representatives of the client. However, this
time the analysts will present their understanding of what they learned at the first meeting and from
studying documentation that has been made available to them by the client. The second step also
involves a rough, high-level security analysis. During this analysis the first threats, vulnerabilities,
threat scenarios and unwanted incidents are identified. They will be used to help with directing and
scoping the more detailed analysis still to come.

**Step 3** The third step involves a more refined description of the target to be analysed, and also all assump-
tions and other preconditions being made. Step three is terminated once all this documentation has
been approved by the client.

**Step 4** This step is organised as a workshop, drawn from people with expertise on the target of the anal-
ysis. The goal is to identify as many potential unwanted incidents as possible, as well as threats,
vulnerabilities and threat scenarios.

**Step 5** The fifth step is also organised as a workshop. This time the focus is on estimating consequences
and likelihood values for each of the identified unwanted inci-dents.

**Step 6** This step involves giving the client the first overall risk picture. This will typically trigger some
adjustments and corrections.

**Step 7** The last step is devoted to treatment identification, as well as addressing cost/benefit issues of
the treatments. This step is best organized as a workshop.

The terminology of CORAS is to a large degree taken from the standard (Standars Australia / Standars
New Zealand 2004). Therefore, the conceptual foundation is to a large degree in accordance to this
standard.

**B.3.5 ISO/IEC 27001 (BS7799-2:2002)**

ISO 27001 is a standard [ISO27001]. The risk assessment concerns a generic requirement that risk
assessment has to be made through a recognized method but no support is provided.

The Risk treatment is a generic recommendation that risk treatment has to be made.

The risk acceptance concerns an indirectly implied through ”statement of applicability”.

This standard is dedicated to a process of certification. It enables the comparison of an information
security management system through a series of controls. This standard does not cover risk analysis or
certification of the Risk Management. Of UK origin, this standard has been adopted by ISO with some
modifications. A certificate granted according to this standard confirms the compliance of an organization
with defined requirements to information security management and a set of security controls.

**B.4 Our methodology**

Here some advantages/drawback points of each methods described below:

OCTAVE is a self-directed approach, meaning that people from an organization assume responsibility
for setting the organization’s security strategy. OCTAVES is a variation of the approach tailored to the limited
means and unique constraints typically found in small organizations (less than 100 people). OCTAVE
is led by a small, interdisciplinary team (three to five people) of an organization’s personnel who gather
and analyze information, producing a protection strategy and mitigation plans based on the organization’s
unique operational security risks. To conduct OCTAVE effectively, the team must have broad knowledge
of the organization’s business and security processes, so it will be able to conduct all activities by itself.

CRAMM is a risk analysis method developed by the British government organization CCTA (Central
Communication and Telecommunication Agency), now renamed the Office of Government Commerce
(OGC). A tool having the same name supports the method: CRAMM. The CRAMM method is rather
difficult to use without the CRAMM tool. The first releases of CRAMM (method and tool) were based on
best practices of British government organizations. At present CRAMM is the UK government’s preferred
risk analysis method, but CRAMM is also used in many countries outside the UK. CRAMM is especially
appropriate for large organizations, like government bodies and industry.

Special Publication 800-series report gives very detailed guidance and identification of what should
be considered within a Risk Management and Risk Assessment in computer security. There are some
detailed checklists, graphics (including flowchart) and mathematical formulas, as well as references that
are mainly based on US regulatory issues.

The main innovations of the CORAS project stem from its emphasis on integrating risk analysis tightly
into a UML and RM-ODP setting, supported by an iterative process, and underpinned by a platform for
tool-integration targeting openness and interoperability.

The standard ISO/IEC 27001 (BS7799-2:2002) does not cover risk analysis or certification of the Risk
Management.

What we need is a directed and detailed approach and NIST800-30 answers in big part to this. To
evaluate the threat against TAS3 components, a methodology based on the book "Threat Modeling"
[SwiderskiSnyder04] has been chosen. This threat modeling is quite similar to NIST800-30 but proposes
deeper analysis that can be directly integrated with software development tool enabling threat modeling
from design to test. Even if both approaches would be suitable, the latter has been chosen for pragmatic
reasons: a European project already worked successfully with this methodology (the Mosquito Project).
Then, to fulfill specificities of collaborative research projects, we slightly changed the approach. Indeed,
threat modeling is generally used to study threats against an application or a system and to verify that
common threat are correctly handled and mitigated. In TAS3, we focus on the threat model of the TAS3
components and are thus confronted to uncommon threats and mitigations.

The following methodology is used in this document:

1. Identify Components
2. Identify sub components and entry points
3. Identify Assets
4. Identify Threat
5. Identify Attacks
6. Identify Mitigation

B.5 Risk Assessment Methodology

For the different section, we propose a list of possible choices. These choices are just given as example,
may be not cover the entire spectrum of threats, attacks, or mitigation.

B.5.1 Identify the Components

The threat model is defined component by component since components can be reused in different
applications and to let project partner work on their own component(s) in parallel.

Here the list of TAS3 components:

1. T3-ACVS: Authorization Credential Validation Service
2. T3-BP-CLIENT: Business Process User Interface
5. T3-BP-MGR: Business Process Administration Interface
6. T3-BP-PEP: Business Process Policy Enforcement Point
7. T3-BP-PIP: Business Process Policy Information Point
8. T3-BP-SMC: Business Process Security Configuration Component
9. T3-BUS-AUD: Audit Event Bus
10. T3-DEL: Delegation Service
11. T3-IDB-ACISIS
12. T3-IDP-SHIB: Shibboleth IDP
13. T3-IDP-SHIB-AGG: Enhancement to Shibboleth IDP to support attribute aggregation
14. T3-LOG-SAWS: Secure Auditing Web Service
15. T3-LOG-WRAP-SAWS: Wrapper Web service around SAWS with extensions
16. T3-OCT: Online Compliance Testing
17. T3-OCT-PLANNER: Test Planner for the Online Compliance Testing
18. T3-ONT-SER: Ontology Server
19. T3-PDP-BP: Business Process Policy Decision Point
20. T3-PDP-BTG: Break the Glass PDP
21. T3-PDP-M: Policy Decision Point/Master PDP
22. T3-PDP-P: PERMIS PDP
23. T3-PDP-T: Trust Policy Decision Point
24. T3-PEP-AI: Application-independent Policy Enforcement Point
25. T3-POL-GUI: Graphical Policy Editor
26. T3-POL-CNL: Controlled Natural Language Policy Editor
27. T3-POL-WIZ: Wizard Policy Editor
28. T3-PORT-JBOSS: JBOSS portal framework
29. T3-REP-FEDORA: TAS3 reference repository
30. T3-REP-JFEDORA: JFEDORA Library for TAS3 Generic Data Format
31. T3-SG-BASE: SOA Gateway Base System
32. T3-SG-WSP: SOA Gateway Web Service Provider
33. T3-SP-CVT: Europass CV Transcoding Web Service
34. T3-SP-MATCHER: Job Profile Matching Service
35. T3-STACK and T3-SSO-ZXID
36. T3-TPN-TB2: Trust Negotiation Module
37. T3-TRU-CTM: Credential based trust service
38. T3-TRU-KPI: Key Performance Indicators
39. T3-TRU-RTM: Reputation Trust Management Service
40. T3-ZXID-LINUX-X86: ZXID library and tools for TAS3 in apache, Java, PHP, and C
41. ZXID-SRC: Sources for the ZXID package

B.5.2 Identify the Sub Component

Each component offers different entry points, i.e. borderlines between the component and the external world that may be under attack. We distinguish two types of entry points: "intended" entry points, i.e. entry point that are visible to other trust domains (WS-Trust interface, etc.) and "internal" entry points that are not directly related to the TAS$^3$ framework (http-level attacks, direct DB access, getting access to OS key store, etc.).

B.5.3 Identify the assets

Each component has assets, i.e. valuable pieces of information or functionalities that have to be protected against attackers. We can distinguish between functional assets, i.e. assets related to the application itself (e.g. medical data in e-health service), and non-functional assets, i.e. assets related to the TAS3 framework itself. This document focuses on the TAS3 components and mainly describes non-functional assets. This is not covered by traditional approaches.

B.5.4 Identify the Threats

A threat is a high-level description explaining what can go wrong (e.g. personal data disclosed, critical data destroyed). It is not directly related to the technology used to implement the component.

1. Auditing and Logging
   - User denies performing an operation
   - Attackers exploit an application without leaving a trace
   - Attackers cover their tracks

2. Authentication
   - Network eavesdropping
   - Brute force attacks
   - Dictionary attacks
   - Cookie replay attacks
   - Credential theft

3. Authorization
   - Elevation of privilege
   - Disclosure of confidential data
- Data tampering
- Luring attacks

4. Configuration Management

- Unauthorized access to administration interfaces
- Unauthorized access to configuration stores
- Retrieval of plaintext configuration secrets
- Lack of individual accountability
- Over-privileged process and service accounts

5. Cryptography

- Poor key generation or key management
- Weak or custom encryption
- Checksum spoofing

6. Exception Management

- Attacker reveals implementation details
- Denial of service
- Sensitive Data
- Access to sensitive data in storage
- Network eavesdropping
- Data tampering

7. Input / Data Validation

- Buffer overflows
- Cross-site scripting
- SQL injection
- Canonicalization
- Query string manipulation
- Form field manipulation
- Cookie manipulation
- HTTP header manipulation

8. Session Management

- Session hijacking
- Session replay
- Man in the middle
B.5.5 Identify the Attacks

An attack is a concrete way to make a threat real (e.g. attacker gets access to personal data). Attacks can be related to the technology used to implement the component. Here some examples of potential attacks.

- Buffer Overflow Attack
- Canonicalization Attack
- Chosen Plaintext Attack
- Cross Site Scripting Attack
- Denial of Service Attack
- Forceful Browsing Attack
- Format String Attack
- HTTP Replay Attack
- Integer Overflow Attack
- LDAP Injection Attack
- Man in the Middle Attack
- Network Eavesdropping Attack
- One-click Attack
- Credentials Brute Force Attack
- Password Dictionary Attack
- Repudiation Attack
- Response Splitting Attack
- Server-side Code Injection Attack
- Session Hijacking Attack
- SQL Injection Attack
- XML Injection Attack

B.5.6 Identify the mitigation

Mitigation is a mechanism to prevent the attack. Mitigations can be features of the framework (e.g. credentials are signed by issuers), can be mechanisms offered by underlying OS or network (e.g. private keys are stored in the machine’s key store) or can be missing (e.g. not implemented). In this last case, the threat is not mitigated. However, it is useful to know that this threat exists. For instance, the fact that non-repudiation is not ensured for message X in component Y can be acceptable for a type of application but when the same component would be reused in another application, mitigation may be necessary.

For mitigation, the following are used:

**Implemented Mitigation** means that the mitigation is implemented and used in the demonstrator.
Existing Mitigation means that there is an existing mitigation that could be used without implementation specific to TAS3. For instance using https or a firewall would mitigate the attack.

Not Implemented Mitigation means that the mitigation has not been implemented. A solution is known but more development is required to use it.

No Mitigation means that there is no solution to mitigate the attack. In this case the attack has to be "acceptable".

When multiple mitigations exist, they are listed in the tree, i.e. by default there is an "or" operator between tree’s nodes where Implemented or :( equals Implemented.

1. Input / Data Validation

   - Assume all input is malicious.
   - Centralize your approach.
   - Do not rely on client-side validation.
   - Be careful with canonicalization issues.
   - Reject known bad input
   - Constrain input.
   - Validate data for type, length, format, and range.
   - Reject known bad input.
   - Sanitize input.
   - Encrypt sensitive cookie state.
   - Make sure that users do not bypass your checks.
   - Validate all values sent from the client.
   - Do not trust HTTP header information.
   - Encrypt sensitive cookie state.
   - Do not trust fields that the client can manipulate (query strings, form fields, cookies, or HTTP headers).
   - Validate all values sent from the client.
   - Do not trust input
   - Consider centralized input validation.
   - Do not rely on client-side validation.
   - Be careful with canonicalization issues.
   - Constrain, reject, and sanitize input.
   - Validate for type, length, format, and range.

2. Authentication

   - Do not use passwords. Use hardware authentication tokens instead.
   - Separate public and restricted areas.
   - Use account lockout policies for end-user accounts.
   - Support password expiration periods.
   - Be able to disable accounts.
   - Do not store passwords in user stores.
   - Use strong passwords.
- Do not send passwords over the wire in plaintext.
- Protect authentication cookies.
- Partition site by anonymous, identified, and authenticated area.
- Support password expiration periods and account disablement.
- Do not store credentials (use one-way hashes with salt).
- Encrypt communication channels to protect authentication tokens.
- Pass Forms authentication cookies only over HTTPS connections.

3. Authorization
- Use multiple gatekeepers.
- Restrict user access to system-level resources.
- Consider authorization granularity.
- Use least privileged accounts.
- Consider authorization granularity.
- Enforce separation of privileges.
- Restrict user access to system-level resources.

4. Auditing and Logging
- Audit and log access across application tiers.
- Consider identity flow.
- Log key events.
- Secure log files.
- Back up and analyze log files regularly.
- Identify malicious behavior.
- Know what good traffic looks like.
- Audit and log activity through all of the application tiers.
- Secure access to log files.

5. Configuration Management
- Secure your configuration store.
- Maintain separate administration privileges.
- Use least privileged process and service accounts.
- Do not store credentials in plaintext.
- Use strong authentication and authorization on administration interfaces.
- Do not use the LSA.
- Secure the communication channel for remote administration.
- Avoid storing sensitive data in the Web space.

6. Sensitive Data
- Do not store database connections, passwords, or keys in plaintext.
- Avoid storing secrets in the Local Security Authority (LSA).
- Retrieve sensitive data on demand.
- Encrypt the data or secure the communication channel.
- Do not store sensitive data in persistent cookies.
- Do not pass sensitive data using the HTTP-GET protocol.
- Avoid storing secrets.
- Secure the communication channel.
- Provide strong access controls on sensitive data stores.

7. Session Management

- Use SSL to protect session authentication cookies.
- Encrypt the contents of the authentication cookies.
- Limit session lifetime.
- Protect session state from unauthorized access. Secure the channel.
- Encrypt the contents of authentication cookies.
- Protect session state from unauthorized access.

8. Cryptography

- Do not develop your own cryptography.
- Use the correct algorithm and correct key size.
- Secure your encryption keys.
- Use tried and tested platform features.
- Keep unencrypted data close to the algorithm.
- Cycle your keys periodically.
- Store keys in a restricted location.
- Exception Management
- Do not leak information to the client.
- Log detailed error messages.
- Catch exceptions.
- Use structured exception handling.
- Do not reveal sensitive application implementation details.
- Do not log private data such as passwords.
- Consider a centralized exception management framework.

B.6 T3-ACVS: Authorization Credential Validation Service

This is part of the T3-PEP-AI component.

B.7 T3-BP-CLIENT: Business Process User Interface

B.7.1 Entry points

i. Intended entry point: User interface
ii. Intended entry point: SSO-related Communication with IdP
iii. Intended entry point : Communication to BP-Engine - Starting process instances, creating and completing tasks
iv. Internal entry point: Policy Store
v. Intended entry point : Decision requests to the T3-PDP-BP
B.7.2 Asset: State of process instances, confidentiality and integrity of process data

i. Ability to start processes

ii. Ability to complete tasks

iii. Access to process data

Threat 1 Unauthorized access to tasks (i.e., viewing or completing tasks).

Attack Session hijacking attack

Mitigation Implemented Usage of SSL to protect session authentication cookies

Mitigation Implemented Limit session lifetime

Threat 2 Phishing attack.

An attacker can present a site similar to the T3-BP-CLIENT to the user and make him confuse this fake site with the real site. It can thus trick him to enter confidential information because he believes a legitimate business process requests the information.

Mitigation Implemented Use SSL certificate

Use an SSL certificate to proof the site’s identity. This should be accompanied by teaching users basic security precautions on the Web.

B.8 T3-BP-DEL: Business Process Delegation Service

B.8.1 Entry points

i. Intended entry point: Interface for re-assigning tasks to another user

ii. Internal entry point: Decision requests to the T3-PDP-BP

iii. Internal entry point: Registration of changed task assignment in the T3-BP-PIP

B.8.2 Asset: Integrity of user-task assignment

The current functionality of the T3-BP-DEL is re-assignment of tasks from one user to another. Normally, requests to do so are authenticated and authorized.

Threat 1 Assignment of tasks caused by other user than the current owner (or an authorized administrator)

Attack Pose as another user when requesting task-reassignment

Existing mitigation Use TAS3 authentication mechanism (identity assertions) to validate requests.

Threat 2 Re-assignment of tasks to non-authorized users

Attack Spoof PDP response

Implemented mitigation PDP and T3-BP-DEL run on the same dedicated machine.

Existing mitigation Encrypt and sign communication with PDP
B.9 T3-BP-ENGINE-ODE: Apache ODE Business Process Execution Engine

B.9.1 Entry points

i. Intended entry point: Process deployment
ii. Internal entry point: Interaction with BP-Client
iii. Internal entry point: Interfaces for communication with PEP

B.9.2 Asset: Executable process models

Process models define the behavior of applications and deal with sensitive information.

Threat 1 Flawed business process with unintended behavior

Attack Unauthorized deployment of process models

Implemented mitigation Password authentication of administrator access to process deployment

Existing mitigation Strong authentication (cryptographic identity assertions)

Not implemented mitigation Fine-grained authorization rules for process models.

Not implemented mitigation Logging of modifications to process models.

B.9.3 Asset: Running process instances (state and data)

The engine executes instances of business processes. Correct application behavior relies on the state and data of these instances.

Threat Stopping running process instances

If running business process instances are stopped, they won’t complete, possibly causing inconsistent behavior, and preventing reaching the application goal.

Attack Unauthorized access to administration interface

Implemented mitigation Password authentication of administrator access to process deployment

Existing mitigation Strong authentication (cryptographic identity assertions)

Not implemented mitigation Fine-grained authorization rules for process models.

Not implemented mitigation Logging of modifications to process models.

B.10 T3-BP-ENGINE-ODE: Apache ODE Business Process Execution Engine

B.10.1 Entry points

i. Intended entry point: Process deployment
ii. Internal entry point: Interaction with BP-Client
iii. Internal entry point: Interfaces for communication with PEP
B.10.2 Asset: Executable process models

Process models define the behavior of applications and deal with sensitive information.

**Threat**  Flawed business process with unintended behavior

**Attack**  Unauthorized deployment of process models

- **Implemented mitigation**  Password authentication of administrator access to process deployment
- **Existing mitigation**  Strong authentication (cryptographic identity assertions)
- **Not implemented mitigation**  Fine-grained authorization rules for process models.
- **Not implemented mitigation**  Logging of modifications to process models.

B.10.3 Asset: Running process instances (state and data)

The engine executes instances of business processes. Correct application behavior relies on the state and data of these instances.

**Threat**  Stopping running process instances

If running business process instances are stopped, they won’t complete, possibly causing inconsistent behavior, and preventing reaching the application goal.

**Attack**  Unauthorized access to administration interface

- **Implemented mitigation**  Password authentication of administrator access to process deployment
- **Existing mitigation**  Strong authentication (cryptographic identity assertions)
- **Not implemented mitigation**  Fine-grained authorization rules for process models.
- **Not implemented mitigation**  Logging of modifications to process models.

B.11 T3-BP-MGR: Business Process Administration Interface

B.11.1 Entry points

i. Internal entry point: Communication with T3-BP-ENGINE-ODE

ii. Intended entry point: Graphical user interface

B.11.2 Asset: Status of business process instances

The T3-BP-MGR will trigger adaption of business process models and especially the migration of running business process instances to a new version of a model.

**Threat**  Unauthorized alteration of business process instances, unauthorized disclosure of process instance status.

**Attack 1**  Spoofing identity of authorized user in communication between T3-BP-MGR and the engine.

**Existing mitigation**  Strong authentication (cryptographic identity assertions)
Attack 2  Illegitimate operations by authorized users.
The user might use a altered client software. If checks only occur at client side, the user might be able to perform unauthorized actions in the engine.

Not implemented mitigation  Perform all necessary checks (also) on server side.
Existing mitigation  Log all activities for audit purposes, limit the people allowed to perform management activities on the engine.

Attack 3  Spyware
An attacker can try to provide the user with an altered client software, which requests the user to authenticate normally, but makes other requests than what the user thinks he has authorized.

Existing mitigation  Control software installation on the user’s computer.
If software installation is not controlled, the user’s system has to be considered insecure anyway, which makes it prone to all sorts of attacks.

B.12  T3-BP-PEP: Business Process Policy Enforcement Point

B.12.1  Entry points
i. Internal entry point: Communication between PEP and PIP
ii. Internal entry point: Communication between PEP and T3-BP-ENGINE-ODE
iii. Internal entry point: Communication with T3-PDP-BP (policy decision requests and responses)

B.12.2  Asset: Correct policy enforcement
Threat 1  A process communicates without policy enforcement

Attack 1  Process models containing direct calls to third party web services, bypassing the PEP
Implemented mitigation  Manual check of process models
Not implemented mitigation  Automatic check of process models
Not implemented mitigation  Integration of the PEP as a layer in the web service stack of the execution engine

Attack 2  Calling the process’ web service interfaces directly, bypassing the PEP
Existing mitigation  Block access from outside using a firewall.
Not implemented mitigation  Integration of the PEP as a layer in the web service stack of the execution engine

Threat 2  Tricking PEP into using a rogue PDP

Attack  Changing PDP address in PEP configuration.
Implemented mitigation  Protect PEP code and configuration from unauthorized alteration using file system permissions.

B.13  T3-BP-PIP: Business Process Policy Information Point

B.13.1  Entry points
i. Internal entry point: Communication between PEP and PIP
ii. Communication with T3-BP-ENGINE-ODE
iii. Communication with PDP
B.13.2 Asset: Correct information for policy enforcement

**Threat 1** Context information about processes not up to date.

The context of a process (current point of execution, assigned users) continuously changes. Up-to-date information is necessary for correct policy enforcement.

**Attack** Interrupt communication between engine and PIP. Implemented mitigation: Run engine and PIP on the same dedicated machine.

This makes it much harder to interrupt communication.

**Existing mitigation** Enforce a time-to-live on information in the PIP, check directly on the engine if TTL is expired.

Threat 2: PIP accepts and provides invalid/untrusted context information.

**Attack** Push incorrect context information to the PIP.

**Existing mitigation** Cryptographically sign messages to the PIP.

**Existing mitigation** Correctly configure trusted sources.

**Existing mitigation** Only accept input from the same host (using a firewall) which only runs BP-related component.

B.14 T3-BP-SMC: Business Process Security Configuration Component

The T3-BP-SMC processes security specifications for business processes (either as annotations or separate specifications) and translates them into business process related security configuration (e.g. business process policies) and enhances business process models with explicit security subprocesses.

B.14.1 Entry points

i. Intended interface: Security configuration user interface

ii. Intended interface: Processing of annotations to business process model security enhancements

iii. Intended interface: Processing of additional business process security configuration

iv. Internal interface: Deployment of security configuration to runtime components.

B.14.2 Asset: Business process security annotations and security

**Threat 1** Inaccurate use of business process security annotations or specifications

**Attack** Process security modeler makes faulty or incomplete security annotations or specifications

**Mitigation** Process security modeler has to be trained for working with business process security annotations and specifications.

**Threat 2** Unauthorized modification of business process security annotations or specifications

**Attack** Unauthorized person changes the security configuration of a business process.

**Not implemented mitigation** User access control for SMC component, for business process models with security annotations and for security specification files.
B.14.3 Asset: Security enhanced business process models and business process security configurations generated by the SMC component

**Threat** Unauthorized modification of security enhanced business process models or security configurations.

**Attack** Unauthorized person changes the security configuration of a business process or the security enhanced business process itself.

**Not implemented mitigation** Secured deployment of business processes and appropriate security configurations from the SMC component to the business process engine (T3-BP-ENGINE-ODE): Webservice calls with signed file transfer.

**Implemented Mitigation** User access control on application and file system level for the business process engine.

B.15 T3-BUS-AUD: Audit Event Bus

B.15.1 Entry points

i. Intended Entry point: Interface with subscription clients

ii. Intended Entry point: Interface with notification clients

iii. Intended Entry point: Interface with update clients

B.15.2 Asset: Channels of audit event type

**Threat** Overload of service creating failure in audit notifications

**Attack** Denial of service

**Not Implemented Mitigation** To prevent a denial of service attack the services calling the audit bus will be limited to specific pre-authenticated TAS$^3$ infrastructure services. Specific terms of behaviour will govern these services and the Audit Bus will be monitored to detect if a service making calls to the bus is in breach of these (an example could be behaviour that can be deemed as a denial of service attack). In this case the offending service will be reported to the appropriate authority and blacklisted from calling the bus. The architecture will also be designed to allow scaling and multiple audit bus implementations to manage load better.

B.15.3 Asset: Status of audit event type

**Threat 1** Unauthorized invocation of the audit bus

**Attack** Unauthorized access to audit bus Web service interface

**Not Implemented Mitigation** In order to update the status of an audit event or invoke the bus a token will be needed. The token is used to present the correct credentials to the audit bus server. This token comes from the main TAS$^3$ authorisation infrastructure.

**Threat 2** Audit bus notification leak

**Attack** Unauthorized access to audit bus notifications
B.16 T3-DEL: Delegation Service

B.16.1 Entry points
i. SOAP web interface
ii. Apache PHP client
iii. Shell or root shell administrative login

B.16.2 Assets
i. Issued credentials (these are cryptographically protected)
ii. Private signing key of the delegation issuing service
iii. Configuration file (list of trusted proxies)
iv. Trust store of the delegation issuing service (application server)
v. The delegation policy (this is cryptographically protected)

B.16.3 Threats
i. Unauthorized delegation
ii. Unauthorized revocation (leading to denial-of-service)
iii. Denial-of-service attack so that authorised requests will be denied

Attack 1: Compromising the private signing key of the service.

A compromise of the signing key of the delegation issuing service could lead to fake credentials being issued, and hence to unauthorized delegations.

Existing Mitigation: Keep the private key of the delegation issuing service in a password protected key store. Use file system permissions to prevent the service configuration file from being read by any entity except the application server running the service. Use an account with limited capabilities (no login shell) to run the application server. If no direct access to the delegation issuing service is required (e.g. when accessing service through a trusted proxy) the application server can be kept behind a firewall.

Attack 2: Listing oneself as a trusted proxy.

By definition, a trusted proxy is trusted to tell the delegation issuing service who the actual issuer is. If one can add oneself to the list of trusted proxies and obtain an SSL certificate that will be trusted by delegation issuing service (or its application server) then one can make false issuing requests (that will be accepted if they obey the delegation policy). Likewise, once the delegation issuing service is fooled into thinking that one is a trusted proxy, one can make bogus revocation requests.

Existing Mitigation: Use file system permission to prevent unauthorized modification of the...
service configuration file (which contains the list of trusted proxies). Alternatively digitally sign the configuration file. Or use a password protected trust store to prevent unauthorized modification of it. Use a reputable CA, or an in house CA (with an off line private key) to issue SSL certificates to the trusted proxies.

**Attack 3** Modifying or replacing the delegation policy.

If one can modify the delegation policy then one can for instance add oneself as a trusted attribute issuer, or add oneself as a member of a specific domain etc.

**Implemented Mitigation** Use a (self-signed) X.509 AC to hold the delegation policy. Use signature verification prior to loading the policy to ensure the integrity of the X.509 AC. Use file system permissions to protect the delegation issuing service’s configuration files from unauthorized modification. This configuration file contains the policy reference, policy location and the certificates that are the roots of trust of the signature verification process.

**Attack 4** Gaining access to the credential repository.

Since the credentials are digitally signed then it is not possible to either modify a credential or mint a new one without access to the signing key. If an attacker gains access to the credential repository she can only delete the credentials in the repository. This will have the effect that legitimate access requests will be denied by a PDP when configured to pull from this repository.

**Existing Mitigation** Restrict access to the credential repository by for instance keeping it behind a firewall. When using an LDAP repository configure it so that the login credentials given write access to the repository cannot be obtained by simply sniffing the network.

**Attack 5** Obtain the password of a legitimate user of the DIS client.

Once an attacker has obtained the password of a legitimate user of the DIS client, then she can make false delegation requests.

**Existing Mitigation** Use an SSL connection to protect the password from being visible on the Internet. Configure the (Apache) web server to not serve the DIS client pages on a http only connection. Ensure that user passwords are strong and impossible to dictionary attack.

### B.17 T3-IDP-ACSIS

#### B.17.1 Entry points

i. Shell or root shell (or ssh) administrative login

ii. TAS3 designed management interfaces (none yet)

iii. Product specific management interfaces

- New user registration (feature to allow anonymous new user to self register)
- Auto-CoT (fully automatic metadata exchange and trust establishment with anonymous third party SPs)
- New service registration (feature to allow anonymous 3rd party to register new services)

iv. Web GUI

v. SOAP web service

- SSO
- SLO
- Discovery query
B.17.2 Data assets

i. Private keys of the service itself

ii. Circle-of-Trust database

iii. Discovery Registrations

iv. User database
   - User names
   - Authentication credentials (password hash, Yubikey shared secret)
   - User’s attribute data

v. Federation database: name id mappings

vi. Session store

B.17.3 Nonfunctional assets

i. Privacy preserving through avoidance of correlation handles

ii. User consent and control of data release

iii. Organizational control of data release

iv. Nonrepudiation

v. Accountability

vi. Credible authentication of users

vii. Credible authentication of system entities

B.17.4 Attacks and mitigation

• Too numerous to describe exhaustively in one afternoon *** TBD

• Generally the data assets are protected using Unix filesystem permissions against shell and local Unix process access. This, of course, is of little value against root. Therefore deployment MUST use nonroot users for running all TAS3 related processes as well as for most administrative tasks.

• The TAS3 designed and product specific management interfaces follow good coding practises (e.g. check for ".." in path) to only allow designed access to the data assets.

• Web GUI is coded such that only authorized accesses are possible

• SOAP web service is coded such that only authorized accesses are possible

• Appropriate crypto layer (such as TLS) is applied in Web GUI, SOAP, and ssh entry points

B.18 T3-IDP-SHIB: Shibboleth IDP

B.18.1 Entry Points

i. SOAP Authentication Service Endpoint

ii. SOAP Attribute Authority Endpoint

iii. Shell or root shell administrative login
B.18.2 Asset: Issued Credentials

**Threat** Issues Credentials used by More than One Party

**Attack** Replay Attack

If intercepted by a third party, credentials may be submitted to the SP at which they are valid multiple times. Without adequate replay protection this may allow multiple users to access the service using the same set of credentials.

**Existing Mitigation** Replay Detection

Implement checks at the SP to log the details of each incoming message so that if the same message is passed to the SP multiple times it can be recognised and the presenter denied access.

---

B.18.3 Asset: Private Signing Keys of the Service

**Threat** Private Signing Key of the Service is Compromised

A compromise of the signing key of the modified IdP could lead to fake attribute or authorisation credentials being issued, and allow an untrusted service to masquerade as the IdP entity.

**Attack** Attacker Gains Access to Machine Hosting the Service

**Existing Mitigation** Use Password Protected Key Store

Keep the private key of the IdP in a password protected key store.

**Existing Mitigation** Use File System Permissions

Use file system permissions to prevent the service configuration file from being read by any entity except the application server running the service.

**Existing Mitigation** Use Restricted Account

Use an account with limited capabilities (no login shell) to run the application server.

**Existing Mitigation** Restrict Access to Service

If no direct access to the IdP is required (e.g. when accessing service through a trusted proxy) the application server can be kept behind a firewall.

---

B.18.4 Asset: User Database (Attributes and Authentication Credentials)

**Threat 1** User Authentication Credentials are Compromised

Once an attacker has obtained the password of a legitimate user of the IdP client, then she can masquerade as the user throughout the federation.

**Attack 1** Dictionary attack

An attacker may attempt to obtain the username / password combination of a legitimate user of the IdP by performing a dictionary attack on the authentication page.

**Implemented Mitigation** Restriction of Number of Authentication Requests

The authentication endpoint should monitor the number of authentication attempts that are made for each incoming authentication session. After a configurable number of failed authentication attempts have been made the authentication attempt should fail and return an error message to the requesting SP.

**Attack 2** Network Sniffing

**Implemented Mitigation** Use SSL/TLS connection

Use an SSL connection to protect the password from being visible on the Internet. Configure the (Apache) web server to not serve the IdP authentication pages on a http only connection.
**Implemented Mitigation** Use Different Authentication Mechanism

Implement two factor or challenge/response authentication mechanisms to increase the user’s level of assurance.

**Threat 2** IdP Assertions are Reused (Credential Theft)

**Attack** Assertions are Stolen

Once issued by the IdP attribute and authorisation credentials may be stolen in transit by an intermediary and presented to another SP in order to grant another user access.

**Implemented Mitigation:** Limited Life Time

Every credential should be short lived to prevent an attacker having unlimited time to reuse the credential.

**Implemented Mitigation:** Restrict SPs at which Assertion is valid

Each credential should be encrypted to the intended recipient making the credential worthless to all but the intended recipient. SPs should obey any audience restriction present in the assertion.

**Implemented Mitigation:** Tie Authentication and Attribute Assertion Together

Each attribute credential should contain a Subject that is identical to that of the Subject of the preceding authentication assertion to prevent it from being used as part of a different session.

**Threat 3** User Attributes are Disclosed to Unauthorised Party

User attributes should not be disclosed to unauthorized parties.

**Attack** Network Sniffing

**Implemented Mitigation** Use of Cryptographic Techniques

All incoming messages are encrypted using the IdPs public key and the IdP encrypts all outgoing messages using the public key of the recipient SP.

**Threat 4** Compromise of Backend Database / LDAP Server

If an attacker can access or modify the backend database/LDAP server then she may be able to edit the credentials.

**Attack** Unauthorised Access Gained to Backend Data Store

**Existing Mitigation** Use of Firewall

Use a firewall to restrict access to the backend data store. This need not be accessible from the Internet, even if the IdP has to be.

**Existing Mitigation** Use of SSL/TLS

When administrator access is needed to the backend data store, then the credentials should not be conveyed in clear text. Communication with the database should either be local or over SSL.
B.18.5 Asset: Session Information

B.18.6 Asset: Configuration Files of the Service

Threat  Unauthorised Modification of the Configuration Files

If an attacker can modify the IdPs configuration files then they can edit the attributes that will be released from the IdP, add additional un-trusted attribute sources, or remove security precautions such as the signing and encryption of outgoing messages.

Attack  Attacker Gains Unauthorised Access and Modifies Configuration Files

Existing Mitigation  Use of File System Permissions

Use file system permissions to prevent the service configuration files from being read by any entity except the application server running the service.

Existing Mitigation  Use of Restricted Account

Use an account with limited capabilities (no login shell) to run the application server.

B.18.7 Non-functional Asset: User Consent and Control of Data Release

B.18.8 Non-functional Asset: Organisation Control of Data Release

B.19 T3-IDP-SHIB-AGG: Enhancement to Shibboleth IDP to support attribute aggregation

Please note: The more general risk analysis presented in preceding Section 2.18 for the Shibboleth IdP also applies to this section. The additional risks presented below apply only to the changes required to support aggregation.

B.19.1 Entry points

i. SOAP web interface for auditing

ii. SOAP web interface for enquiries

iii. Administrative shell login

iv. Application management console of application server

B.19.2 Asset: The Audit Trail

Threat  Modification of Audit Trail

Attack  Attacker Accesses Auditing Service Machine and Modifies/Deletes the Audit Trail

An attacker may gain unauthorized access to the machine hosting the logging service. In this way she may be able to modify or remove the audit trail.

Implemented Mitigation  Cryptographic Protection

The SAWS audit trail structure uses cryptographic techniques to ensure that unauthorized modification of the audit trail will not go unnoticed. It uses heartbeats to ensure that records are not removed from the end. It chains audit files together to ensure that complete audit files cannot be deleted without notice. It periodically creates new audit files to ensure they do not become too long.
Existing Mitigation  File System Permissions
Use file system permissions to restrict access to the audit trail, e.g. only allow the application server’s account to write to the file.

Existing Mitigation  Use Restricted Account
Run the application server with a restricted account (e.g. an account without a login shell). Restrict access to the machine hosting the logging service by e.g. running it behind a firewall if possible.

Existing Mitigation  Backup the Audit Trail
Periodically backup complete audit files to CD-ROM or similar to prevent loss or modification in case of an attack.

Possible Threats

Threat 2 Unauthorized Disclosure of Audit Trail Contents

Attack 1  Attacker Obtains Complete Copy of Audit Trail
If the machine hosting the audit service (or one of the machines holding the backup of the audit trail) is compromised then it may be possible that the attacker gets to read access to audit information that she shouldn’t be able to read.

Implemented Mitigation  Encryption of Audit Trail Data
If confidentiality of the audited data is important then the audit service can be configured to encrypt the audited data with a symmetric key. This symmetric key is stored in the audit trail, encrypted with the public key of the persons authorised to read the contents of the trail. In this way, the audit trail is not readable by subjects not having one of the correct private keys.

Attack 2  Attacker Abuses the Enquiry Service
An attacker may abuse the enquiry service by issuing search requests for data that she isn’t allowed to see.

Existing Mitigation  Use Authorisation on Enquiry Service
The enquiry service is still in an embryonic state of implementation, but it is envisaged that a PDF will be used both before searching the audit trail and to filter the results returned to the requester. The enquiry service will run over SSL and will require client authentication, possibly using a list of trusted proxies to say who the actual requester is.

B.19.3 Asset: SAWS Records a Secure Audit Trail

Threat  Denial of Service for Legitimate Clients

Attack  Attacker Swamps Service with Requests

Implemented Mitigation  Only Allow Known Clients
The auditing service only accepts messages from known clients. This is ensured by configuring the application server to run the auditing service over SSL only. The application server has to be configured to require client authentication. Configure the trust store to only accept messages from clients having a certificate issued by a trusted CA. Further, the SAWS configuration file contains a list of known clients; only these are allowed to add messages to the audit trail.

Existing Mitigation  Use Firewall
When the IP-addresses of the attacker are known these can already be blocked on the level of the (operating-system) firewall, so that they never reach the application server.

B.19.4 Asset: Private Keys for Signing/Encrypting

Threat  Compromise of the Private Signing or Encryption Key
**Attack**  Attacker Gains Access to Machine Hosting Auditing Service

An attacker may gain access to the machine hosting the audit service and in this way obtain a copy of service’s secrets. For instance, if a copy of the private signing key is obtained, then an attacker would be able to modify the audit trail without this being noticed.

**Implemented Mitigation**  Use of Password Protected Key Stores

Use password protected key stores (which can optionally require more than one password to be opened) to hold the service’s secrets. The passwords are asked for interactively and are not stored in the configuration file. Only when automatic (i.e. without human intervention) start up of the service is required do you need to store passwords in a file. This file should be properly protected using file system permissions and should only be readable by the application server hosting the audit service.

**Existing Mitigation**  Use of TPM to Store Secrets

The encrypted software keystore could be replaced with a hardware TPM based keystore which will forbid tampering or unauthorised access.

**Existing Mitigation**  File System Permissions

---

**B.20**  T3-LOG-SAWS: Secure Auditing Web Service

**B.20.1**  Entry points

i. SOAP web interface for auditing

ii. SOAP web interface for enquiries

iii. Administrative shell login

iv. Application management console of application server

**B.20.2**  Asset: The Audit Trail

**Threat 1**  Modification of Audit Trail

**Attack**  Attacker Accesses Auditing Service Machine and Modifies/Deletes the Audit Trail

An attacker may gain unauthorized access to the machine hosting the logging service. In this way she may be able to modify or remove the audit trail.

**Implemented Mitigation**  Cryptographic Protection

The SAWS audit trail structure uses cryptographic techniques to ensure that unauthorized modification of the audit trail will not go unnoticed. It uses heartbeats to ensure that records are not removed from the end. It chains audit files together to ensure that complete audit files cannot be deleted without notice. It periodically creates new audit files to ensure they do not become too long.

**Existing Mitigation**  File System Permissions

Use file system permissions to restrict access to the audit trail, e.g. only allow the application server’s account to write to the file.

**Existing Mitigation**  Use Restricted Account

Run the application server with a restricted account (e.g. an account without a login shell). Restrict access to the machine hosting the logging service by e.g. running it behind a firewall if possible.

**Existing Mitigation**  Backup the Audit Trail

Periodically backup complete audit files to CD-ROM or similar to prevent loss or modification in case of an attack.
Threat 2 Unauthorised Disclosure of Audit Trail Contents

**Attack 1** Attacker Obtains Complete Copy of Audit Trail

If the machine hosting the audit service (or one of the machines holding the backup of the audit trail) is compromised then it may be possible that the attacker gets to read access to audit information that she shouldn’t be able to read.

**Implemented Mitigation** Encryption of Audit Trail Data

If confidentiality of the audited data is important then the audit service can be configured to encrypt the audited data with a symmetric key. This symmetric key is stored in the audit trail, encrypted with the public key of the persons authorised to read the contents of the trail. In this way, the audit trail is not readable by subjects not having one of the correct private keys.

**Attack 2** Attacker Abuses the Enquiry Service

An attacker may abuse the enquiry service by issuing search requests for data that she isn’t allowed to see.

**Existing Mitigation** Use Authorisation on Enquiry Service

The enquiry service is still in an embryonic state of implementation, but it is envisaged that a PDP will be used both before searching the audit trail and to filter the results returned to the requester. The enquiry service will run over SSL and will require client authentication, possibly using a list of trusted proxies to say who the actual requester is.

B.20.3 Asset: SAWS Records a Secure Audit Trail

**Threat** Denial of Service for Legitimate Clients

**Attack** Attacker Swamps Service with Requests

**Implemented Mitigation** Only Allow Known Clients

The auditing service only accepts messages from known clients. This is ensured by configuring the application server to run the auditing service over SSL only. The application server has to be configured to require client authentication. Configure the trust store to only accept messages from clients having a certificate issued by a trusted CA. Further, the SAWS configuration file contains a list of known clients; only these are allowed to add messages to the audit trail.

**Existing Mitigation** Use Firewall

When the IP-addresses of the attacker are known these can already be blocked on the level of the (operating-system) firewall, so that they never reach the application server.

B.20.4 Asset: Private Keys for Signing/Encrypting

**Threat** Compromise of the Private Signing or Encryption Key

**Attack** Attacker Gains Access to Machine Hosting Auditing Service

An attacker may gain access to the machine hosting the audit service and in this way obtain a copy of service’s secrets. For instance, if a copy of the private signing key is obtained, then an attacker would be able to modify the audit trail without this being noticed.

**Implemented Mitigation** Use of Password Protected Key Stores

Use password protected key stores (which can optionally require more than one password to be opened) to hold the service’s secrets. The passwords are asked for interactively and are not stored in the configuration file. Only when automatic (i.e. without human intervention) start up of the service is required do you need to store passwords in a file. This file should be properly protected using file system permissions and should only be readable by the application server hosting the audit service.
Existing Implementation  Use of TPM to Store Secrets
The encrypted software keystore could be replaced with a hardware TPM based keystore which will forbid tampering or unauthorised access.

Existing Mitigation  File System Permissions

B.21  T3-LOG-WRAP-SAWS: Wrapper Web service around SAWS with extensions

B.21.1  Entry points
i.  SOAP client

B.21.2  Asset: Audit Trails

Threat 1  Traceless Exploitation
Attack  Attackers exploit an application without leaving a trace
Mitigation  Identify malicious behaviour.

Threat 2  Disguising the attack
Attack  Attackers cover their tracks
Existing Mitigation  Using audit/log files
Existing Mitigation  Audit and log activity through all of the application tiers.
Existing Mitigation  Secure access to log files.

B.22  T3-OCT: Online Compliance Testing
The TAS³ architecture supports a novel testing approach for services that are running within a TAS³ choreography. Such approach is implemented within the TAS³ architecture by the OCT (On-line Compliance Testing) component. The novel idea behind OCT is that services are submitted to the execution of a set of test cases in their real execution environment, while they possibly interact with other services. The service under tests are unaware that some invocation they receive comes from OCT. OCT validation may be performed periodically or after some relevant event, aiming at verifying that every service in the TAS³ federation abides by the manifested policies and/or complies with the functional specifications. As detailed in D10.2, on-line testing in TAS³ is applied in order to reduce the risk that services within a circle of trust (i.e. service federation) will get in contact with unreliable services. Therefore services within the federation will be regularly submitted to on-line testing to assess that they comply with their public and manifested policies. Also, the scenario we foresee for OCT assumes that, within the federation, services admit that different requesters may play different roles. Service clients collect credentials in order to prove the role that they are playing as requesters. Thus, the authorization/authentication layers in TAS³ (e.g. the IdP components) collaborates with OCT components signing role assertions that would be used in running the test cases.

Note that the OCT components uses the security layer provided by the TAS³ architecture. Thus it exists a strict dependency in term of security vulnerabilities with the issues described for the T3-STACK and the component T3-SSO-ZXID (see Section 4.39).

B.22.1  Entry points
i.  Intended Entry Point: SOAP Binding with the IdP
ii.  Internal Entry Point: The API of the OCT component, used in order to implement the test-driver
B.22.2 Non-Functional Assets

i. the OCT component credentials

ii. user/test user credentials

iii. QoS of the service under test

The following threats are relate to the non functional assets

Threat 1 The service under test have to be unaware of the TAS3 testing session, the credential issued for testing purposes can be actually spent as valid credential within the federation. Malicious services may obtain from the IdP credential that was originally issued only for testing purposes and use them illicitly.

Attack 1 Man in the middle
Malicious services can sniff the communication between the OCT component and the IdPs in TAS3

Attack 2 Spoofing
Malicious services masquerade their interaction to the IdP as the OCT component.

Implemented Mitigation:

i. Usage of SSL or TLS Client Certificate to protect session authentication in the communication between the OCT component and the IdPs

ii. Use the security assets of the TAS3-STACK

iii. Limit session lifetime

Threat 2 A massive use of the on-line testing process can affect the QoS of the service under test.

Attack Denial of Service (DoS)
Malicious services that have gained the OCT credential can exploit them running Denial-Of-Services attacks to other services in the TAS3 choreography

Implemented Mitigation Use the security assets of the TAS3-STACK

Threat 3 The credential used during an on-line testing session may have undesired effect on actual users.

The threat 4.18.2.5 does not refer to any specific attack. However, as potential errors and failures of the on-line testing activity may generate unconsidered security leak affecting real user, we recommended some implemented mitigations for this threat

Implemented Mitigation Create and use realistic test user

B.23 T3-OCT-PLANNER: Test Planner for the Online Compliance Testing

The TAS3-OCT-PLANNER is an off-line component. With respect to the TAS3 Architecture it belongs to the Modeling & Configuration Management Realm. In particular, the TAS3-OCT-PLANNER can be used off-line in order to model test cases, and it is not affected by run-time security issues.

B.24 T3-PDP-BP: Business Process Policy Decision Point

The need of this component is currently not clear; see PDP-M and PDP-P for possible attacks;
B.25 TAS3-ONT-SER: Ontology Server

B.25.1 Entry Points

- Command line access
- J2EE Interface: uses JBoss, which provides means for security (authentication, etc.)

Intended Entry point:

i. J2EE Bean delegating calls to internal beans
ii. LEXONBASE (CRUD LEXON & CONTEXT)
iii. COMMITMENT (CRUD COMMITMENT)
iv. VERSIONNING (prototype)
v. PERSPECTIVE CONFLICT MANAGEMENT (prototype)

B.25.2 Asset: Ontology Server

Threat Trust Policy Disclosure/Modification (heading4)

**Attack 1** The administrator access to the repository and disclose/change the trust policies

**Existing Mitigation** External Monitoring and Logging mechanism
Standard provided by JBoss for the application server WEB CONSOLE + logging via shell Keeping of who modified what in the business layer

**Attack 2** Unauthorized access to the ontology server

**Existing Mitigation** Logging and monitoring mechanisms
The data assets are stored in a Postgres database on a Linux server, which is protected through the login to the database.
The J2EE interface uses username and passwords to enable authorised access to the data.
The REST interface will be implemented and will give authorised access to people with the right credentials/tokens.

**Existing Mitigation: Use of authorization mechanisms**

- provided by JBoss
- IP blocked for 10 minutes after 3 unsuccessful logins (prevent intrusion)
- no open ports

**Attack 3** forcing the access to the database

**Existing Mitigation** Use of secure channels for data transmissions (i.e. HTTPS)

B.26 T3-PDP-BTG: Break the Glass PDP

This is part of the T3-PEP-AI component. Here we discuss an additional asset specific to the T3-PDP-BTG

B.26.1 Entry points

See entry points of T3-PEP-AI.
B.26.2 Asset: The BTG-State of the System

**Threat** Unauthorised Modification of the BTG-State

When an attacker manages to modify the BTG state, then she may be able to get access to certain resources without actually breaking the glass, hereby circumventing any obligations (such as audit and notification) that may be associated with breaking the glass.

**Attack** Gain Access to Machine Hosting BTG-State and Modify It

**Existing Mitigation** In-Memory Only Implementation

The BTG-state is implemented in-memory only, making it hard for an attacker to modify it as the operating system manages the boundaries between the memory areas assigned to different applications.

B.27 T3-PDP-M: Policy Decision Point/Master PDP

This is part of the T3-PEP-AI component.

B.28 T3-PDP-P: PERMIS PDP

B.28.1 Entry Points

i. Java API

B.28.2 Asset: Authorisation Policy

**Threat** Policy is Modified

**Attack** Attacker Modifies Authorisation Policy

An attacker tries to edit the authorisation policy in order to grant access to people who should not have access or deny access to people who should have access.

**Implemented Mitigation** Cryptographic Protection

The PERMIS policy is designed to be digitally signed and the PERMIS engine validates the signature at start up time. Hence an attacker would need to gain access to the private key of the policy author in order to effectively edit the policy.

B.29 T3-PDP-T: Trust Policy Decision Point

B.29.1 Entry points

i. Internal Entry point: Shell or root shell

ii. Intended Entry point: Trust PDP Evaluation Request

iii. Intended Entry point: Trust PDP Call-back Function

iv. Intended Entry point: Returned value from Trust Service call

v. Intended Entry point: Trust Policy Repository Interfaces
### B.29.2 Asset: Trust Policy

**Threat** Trust Policy Disclosure/Modification

A malicious agent accesses the trust policies and discloses or manipulates them. Like any policies, manipulation needs to be prevented and the policies are sensitive information that should be protected from being revealed.

**Attack** Malicious administrator

A malicious administrator accesses to the repository and discloses/changes the trust policies.

**Existing Mitigation** External monitoring and logging mechanisms to track the administrator’s activities. These mechanisms should be external to the Trust-PDP in order to avoid their manipulation performed by the administrator.

**Attack** Unauthorized access

A malicious agent gains access to the trust policy repository to disclose/change the trust policies.

**Existing Mitigation** Logging and monitoring mechanisms to track the access records and to detect illegal behaviours.

**Existing Mitigation** Use of authorization mechanisms to control the access to the trust policy repository.

**Attack** Man in the Middle or network eavesdropping

A malicious agent gains the access to the trust policies during the querying/updating process.

**Existing Mitigation** Use of secure channels for data transmissions (i.e. HTTPS).

### B.29.3 Asset: Trust-PDP Evaluation Service

**Threat 1** Trust-PDP service not available

A malicious agent makes the Trust-PDP unavailable for the intended users.

**Attack** Denial of Service (DOS) attack

A malicious agent or a collectiveness of malicious agents saturates the Trust-PDP evaluation service with a huge amount of requests.

**Implemented Mitigation** The existence of monitoring and logging mechanisms to trace suspect behaviours.

**Existing Mitigation** The utilization of a firewall for the input/output traffic filtering.

**Existing Mitigation** Geographically restrictions on access domain: the service is made available only to well known locations where requests are expected from.

**Threat 2** Unauthorized Access to the Trust PDP evaluation service

A malicious agent uses the Trust-PDP evaluation service, sending requests and obtaining responses to use for malevolent purposes.

**Attack** Brute Force or Password Dictionary

A malicious agent obtains the access performing a brute force attack (trying all the possible passwords) or a password dictionary attack (trying with the most likelihood passwords).

**Existing Mitigation** Mechanisms that press for the usage of strong and effective passwords.
Implemented Mitigation  Logging and monitoring mechanisms to detect possible misbehaviours.

Implemented Mitigation  Single Sign On (SSO) authorization mechanisms to automatically
and safely generate strong and secure tokens.

Threat 3 Request/Response Modification
A malicious agent changes the request or the response in order to manipulate the results of the evaluation.

Attack  Parameter Tampering or Man in the Middle.
A malicious agent gains the access to the messages exchanged during a request/response process and alters such messages in a malevolent way.

Existing Mitigation  Data integrity check mechanisms to verify that a response is feasible for a given request.

Existing Mitigation  Mechanisms to link the response message to its associated request message.

Existing Mitigation  Use of secure channels for data transmissions (i.e. HTTPS)

B.30  T3-PEP-AI: Application-independent Policy Enforcement Point

B.30.1  Entry Points
i. SOAP web interface
ii. Administrative shell login

B.30.2  Asset: Built-in Policies

Threat  Unauthorised Modification of Built-in Policies

Attack  Attacker manages to replace the policy contents of a built-in policy
If an attacker gains access to the machine hosting the policies (which is not necessarily the same as the machine hosting the AIPEP) then maybe she can replace a built-in policy with one of her own making.

Implemented Mitigation  Cryptographic Protection
For PERMIS policies this attack is defended against by encapsulating the policy in a (self signed) X.509 Attribute Certificate. The cryptographic signature on the X.509 AC prevents tampering with its contents.

Existing Mitigation  File System Permissions
For other policy formats, in particular XACML policies for the Sun XACML PDP or for the trust PDP, no such cryptographic protection exists at the moment. File system permissions can give some protection: the policies should only be readable (and not even writeable) by the AI-PEP itself.

B.30.3  Asset: Configuration File

Threat  Unauthorised Modification of Configuration File

Attack  Attacker Modifies Configuration File
If an attacker gains access to the machine hosting the AI-PEP, then she may be able to modify the configuration file e.g. to load an additional policy or to remove a policy which would deny her access.

The configuration file also contains the roots of trust for signature verification, so the attacker might also able to add a PKC of her liking to as a root of trust.
**Existing Mitigation**  File System Permissions

Use file system permissions to protect the configuration file from being changed. Configuration file should only be readable by account running the AI-PEP service.

**Existing Mitigation**  Use Restricted Account

The AI-PEP service should be run under an account with very few permissions, e.g. without remote login capabilities.

**B.30.4 Asset: Roots of Trust (PKCs) for Signature Verification**

**Threat**  Attacker Adds/Deletes/Replaces the PKCs used for Signature Verification

**Attack**  Attacker Gains Access to Machine with PKCs and Modifies Them

**Existing Mitigation**  File System Permissions

**Existing Mitigation**  Use a Password Protected Trust Store

Keeping the PKCs in a password protected trust store requires the attacker to guess the password for the trust store (or otherwise break its security).

**Existing Mitigation**  Use Restricted Account

**B.30.5 Asset: System Coordinates Decision Making**

**Threat**  System Crushes

**Attack**  An Attacker Crafts a Malicious Message in Order to Crash the System

An attacker might attempt to send an authorization request (or a request to validate some credentials) to is formed in such a way as to make the server crash.

**Implemented Mitigation**  Use of Schema Checking

The system is built in such a way that only requests conforming to the schema ever reach the server. A SOAP message not conforming to the schema is discarded by the Axis2 SOAP layer around the actual server.

**B.30.6 Asset: The Sticky Store**

**Threat**  Unauthorised Modification of Sticky Store

Since the sticky store holds the binding between resources and the policies applicable to them, modifying this store might get an attacker unauthorised access to resources.

**Attack**  Attacker Modifies the Sticky Store by Gaining Access to the Machine Hosting the Sticky Store

**Non existing Mitigation**  Use Proper Access Control to Sticky Store

The sticky store hasn’t been fully implemented yet. A first version will be a simple in-memory implementation of the sticky store which is by definition hard to modify for an attacker. However, once a persistent implementation (e.g. by using a relational database) is used then proper access controls need to be in place to prevent unauthorised access to this persistent storage.

**B.31 T3-POL-GUI: Graphical Policy Editor**

**B.31.1 Entry points**

Graphical User Interface of the application
B.31.2 Asset: Information in Configuration File

**Threat** Credentials to Policy Repository are Leaked

**Attack 1** Attacker Reads Configuration File

**Existing Mitigation** File System Permissions

**Non existing Mitigation** Don’t Store Secrets in the Configuration File

**Attack 2** Attacker Sniffs Network Communication

**Existing Mitigation** Use SSL/TLS for Communication between PE and Repository

B.31.3 Asset: Private Signing Key Used to Sign Policies

**Threat** Compromise of the signing key

**Attack** Attacker Gains Access to Machine Running PE and Obtains Signing Key

**Implemented Mitigation** Use a Password Protected Key Store for the Private Key

**Implemented Mitigation** Do Not Store Password in a File

B.32 T3-POL-CNL: Controlled Natural Language Policy Editor

This is the same application as T3-POL-GUI.

B.33 T3-POL-WIZ: Wizard Policy Editor

This is the same application as T3-POL-GUI.

B.34 T3-PORT-JBOSS: JBOSS portal framework

The JBoss portal is used to display some portlets that embed different applications of TAS3 and its partners. For example it is possible to choose the audit viewer-portlet (TAS3-component) and a business process-portlet (later from a TAS3-associate).

The applications itself run on the server of its provider. The JBoss portal only shows the view on the applications. Later on these portlets will be able to communicate with each other e.g. to exchange authentication information.

The JBoss server and the underlying application stack (apache, tomcat, java) is an open source product.

B.34.1 Entry points

**Intended Entry points**

i. WebGui to access the JBoss portal

ii. Idp to get user authentication

iii. Portlets (to show the information of the external applications)

**Internal Entry points**

i. Shell or root shell (or ssh) for administrative login

ii. Tomcat management interface

iii. JBoss portal management interfaces
B.34.2 Asset: user data (functional data) and user credentials (non-functional data)

Threat 1 catch of user data

**Attack**  Man-in-the-Middle-attack
Sniffing and/or modifying the communication between the JBoss portal and the user or other TAS3 components. e.g. catching of passwords or user data.

**Existing Mitigation** end-to-end-encryption
End-to-end-encryption of the whole communication between the TAS3-components, the user and the JBoss portal could prevent a Man-in-the-middle-attack.

Threat 2 Application Stack
To run JBoss or a fedora-repository several additional underlying components are needed. This is e.g. the operating system, ssh, apache, tomcat and the application (Jboss or fedora) itself.

**Attack 1** Flaws in the software components itself
Flaws in the software components (e.g. buffer overflow) enable attackers to obtain more rights than they used to have.

**Existing Mitigation** Automatically installed updates prevent older versions and flaws

**Attack 2** Misconfiguration, e.g. guestuser with administrative rights

**Existing Mitigation** continuous control and tests with security-tools

**Attack 3** easy/bad passwords
Easy passwords e.g. "test" or "secure" can be hacked very easily. This can give an attacker the identity and the rights of the user.

**Existing Mitigation** use of blacklists and password rules

Blacklists can be used to prevent trivial passwords and strong password rules ensure secure passwords.

**Attack 4** careless user management
Careless user management (e.g. expired users are not removed) allow users to have more rights than they should have.

**Existing Mitigation** strict and clear rules and processes for TAS3 partners and members
Continuous checks by the administrators and clear and strict processes for internal user management can support the user management. Rules and processes should be part of the legal documents for TAS3 partnership / member - registration.

Threat 3 compromised other TAS3-components

**Attack 1** compromised idp
A compromised idp could concede wrong data and additional rights to an user.

**Existing Mitigation** ensure Trustworthiness
IdPs and other security-relevant components and partners have to be trustworthy and controlled internally and regularly

**Attack 2** toxic portlets
Portlets could contain unwanted functionality (e.g. additional user request for user id and password that is locally stored --> password phishing) that could allow a Misuse of data.
Existing Mitigation mandatory previous checks

Portlets should be checked mandatory before used in the context of TAS3. Also changes made to the portlets should be notified automatically.

**Attack 3** DoS-attack

Due to Denial-of-Service-attacks a user could be redirected to specific and perhaps modified servers and service providers.

**Existing Mitigation** Continuous monitoring of availability, continuous checks of log files

### B.34.3 Asset: internal data (non-functional data)

**Threat 1** catch of internal data

**Attack** Man-in-the-Middle-attack

Sniffing and/or modifying the communication between the JBoss portal and the administrators or other TAS3 components. e.g. catching of passwords or user data.

**Existing Mitigation** End-to-end-encryption

End-to-end-encryption of the whole communication between the TAS3-components, the user and the JBoss portal could prevent a Man-in-the-middle-attack.

**Threat 2** Application Stack

To run JBoss or a fedora-repository several additional underlying components are needed. This is e.g. the operating system, ssh, apache, tomcat, and the application (Jboss/fedora) itself.

**Attack 1** Flaws in the software components itself

Flaws in the software components (e.g. buffer overflow) enable attackers to obtain more access the server at system level.

**Existing Mitigation** Automatically installed updates prevent older versions and flaws

**Attack 2** Misconfiguration, e.g. guestuser with administrative rights

**Existing Mitigation** continuous control and tests with security-tools

**Attack 3** easy/bad passwords

Easy passwords e.g. "test" or "secure" can be hacked very easily. This can give an attacker the identity and the rights of the user.

**Existing Mitigation** use of blacklists and password rules

Blacklists can be used to prevent trivial passwords and strong password rules ensure secure passwords. This should be fixed especially for administrators.

**Attack 4** careless user management

Careless user management (e.g. expired users are not removed) allow users/admins to access the system even if they should not be allowed any more.

**Existing Mitigation** strict and clear rules and processes for TAS3 partners and members

Continuous checks by the administrators and clear and strict processes for internal user management can support the user management. Rules and processes should be part of the legal documents for TAS3 partnership / member - registration. Especially when a admin is leaving the account has to be closed and the system should be checked for backdoors.

**Threat 3** compromised other TAS3-components

**Attack 1** compromised idp

A compromised idp could concede wrong data and additional rights to an user.
Existing Mitigation ensure Trustworthiness
IdPs and other security-relevant components and partners have to be trustworthy and controlled internally and regularly.

Attack 2 toxic portlets
Portlets could have access to internal data and pass that to an external receiver.

Existing Mitigation mandatory previous checks
Portlets should to be checked mandatory before used in the context of TAS3. Also changes made to the portlets should be notified automatically.

Attack DoS-attack
Due to Denial-of-Service-attacks a user could be "redirected" to specific and perhaps modified servers and service providers.

Existing Mitigation Continuous monitoring of availability, continuous checks of log files

B.35 T3-REP-FEDORA: TAS3 reference repository
UNIKOLD
The Fedora repository is used as a reference repository. It stores user specific data and retrieves it upon confirmed and authorized (service) requests.

The repository itself runs on the server of its provider. The fedora repository and the underlying application stack (apache, tomcat, java) is an open source product. It is enriched by separately programmed modules for TAS3.

B.35.1 Entry points

Intended Entry points

• web service interface to access the repository data

Internal Entry points

i. Shell or root shell (or ssh) for administrative login

ii. Tomcat management interface

iii. fedora repository management interface

iv. Idp to get user authentication

v. PDP to verify the authorization

B.35.2 Asset: user data (functional data) and credentials (non-functional data)

4.31.2.1. Threat catch of user data

4.31.2.1.1. Attack Man-in-the-Middle-attack

Sniffing and/or modifying the communication between the fedora-server and other TAS3 components.

E. G. catching of credentials or user data.

4.31.2.1.1.1. Existing Mitigation::end-to-end-encryption

End-to-end-encryption of the whole communication between the TAS3-components, the user and the fedora-repository could prevent a Man-in-the-middle-attack.
4.31.2.2. **Threat** Application Stack

To run a fedora-repository several additional underlying components are needed. This is e.g. the operating system, ssh, apache, tomcat, java, dbxml, database and the application (Jboss or fedora) itself.

4.31.2.2.1. **Attack** Flaws in the software components itself

Flaws in the software components (e.g. buffer overflow) enable attackers to obtain more rights than they used to have.

4.31.2.2.1.1. **Existing Mitigation** Automatically installed updates prevent older versions and flaws

4.31.2.2.2. **Attack** Misconfiguration

E.g. guestuser with administrative rights

4.31.2.2.2.1. **Existing Mitigation** continuous control and checks with security-tools

4.31.2.3. **Threat** compromised other TAS3-components

4.31.2.3.1. **Attack** compromised PDP

A compromised PDP could concede wrong requests.

4.31.2.3.1.1. **Existing Mitigation** ensure Trustworthiness

PDPs and other security-relevant components and partners have to be trustworthy and controlled internally and regularly

4.31.2.3.2. **Attack** DOS-attack

Due to Denial-of-Service-attacks a valid request could be blocked before it reaches the repository or the answer of the repository could be blocked.

4.31.2.3.2.1. **Existing Mitigation** Continuous monitoring of availability, continuous checks of log files

B.35.3 Asset: internal data (non-functional data)

**Threat** catch of internal data

4.31.3.1.1. **Attack** Man-in-the-Middle-attack

Sniffing and/or modifying the communication between the Fedora repository and the adminstrators or other TAS3 components. E.g. catching of passwords or user data could be a security problem.

4.31.3.1.1.1. **Existing Mitigation**: end-to-end-encryption

End-to-end-encryption of the whole communication between the TAS3-components, the user and the Fedora repository could prevent a Man-in-the-middle-attack.

4.31.3.2. **Threat** Application Stack

To run a fedora-repository several additional underlying components are needed. This is e.g. the operating system, ssh, apache, tomcat, java, dbxml, a database and the fedora repository application itself.
4.31.3.2.1. **Attack** Flaws in the software components itself

Flaws in the software components (e.g., buffer overflow) enable attackers to obtain access to the server at system level.

4.31.3.2.1.1. **Existing Mitigation** Automatically installed updates prevent older versions and flaws

4.31.3.2.2. **Attack** Misconfiguration

- e.g. guestuser with administrative rights

4.31.3.2.2.1. **Existing Mitigation** continuous control and tests with security-tools

4.31.3.2.3. **Attack** easy/bad passwords

- Easy passwords e.g. "test" or "secure" can be hacked very easily. This can give an attacker the identity and the rights of the user.

4.31.3.2.3.1. **Existing Mitigation** use of blacklists and password rules

- Blacklists can be used to prevent trivial passwords and strong password rules ensure secure passwords. This should be fixed especially for administrators.

4.31.3.2.4. **Attack** careless user management

- Careless user management (e.g., expired users are not removed) allow users/admins to access the system even if they should not be allowed any more.

4.31.3.2.4.1. **Existing Mitigation** strict and clear rules and processes for TAS3 partners and members

- Continuous checks by the administrators and clear and strict processes for internal user management can support the user management. Rules and processes should be part of the legal documents for TAS3 partnership / member - registration. Especially when a admin is leaving the account has to be closed and the system should be checked for backdoors.

4.31.3.3. **Threat** compromised other TAS3-components

4.31.3.3.1. **Attack** DOS-attack

- Due to Denial-of-Service-attacks a user could be "redirected" to specific and perhaps modified servers and service providers.

**Existing Mitigation** Continuous monitoring of availability, continuous checks of log files

**B.36 T3-REP-JFEDORA: JFEDORA Library for TAS3 Generic Data Format**

**B.36.1 Entry points**

- Java (JAR) library interface
B.36.2 Asset: Message based on Generic Data Format

Threat  Exploitation of messages

Attack  Attackers exploit an application without leaving a trace

Existing Mitigation  Identify malicious behavior.

Attack  Attackers cover their tracks

Existing Mitigation  Audit and log activity through all of the application tiers.
Existing Mitigation  Secure access to log files

B.37 T3-SG-BASE: SOA Gateway Base System

B.37.1 Entry points

i. Internal Entry point: HTTP PUT/DELETE Method for specific files in configuration directory

ii. Internal Entry point: Shell login to machine for administration

iii. Intended Entry point: A SOAP web service call via the T3-SG-WSP component

B.37.2 Asset: Backend Legacy data

Threat  SOA Gateway not available

Attack  DOS Attack

Existing Mitigation  Monitoring and Logging mechanisms
Existing Mitigation  Firewalls utilization

Implemented Mitigation  Reject requests not validated by T3-SG-WSP component.

Threat  Unauthorized Access to the SOA Gateway services

Attack  Read/update data using SOA Gateway services

Existing Mitigation  Require use of HTTPS

Implemented Mitigation  Reject requests not validated by T3-SG-WSP component.

Implemented Mitigation  Ensure TAS3 security is always enabled on the SOA Gateway.

B.37.3 Asset: Server Configuration Files

Threat  Gain SSH access to machine

Attack  Once SSH access is granted, the attacker has access to sensitive data.

Implemented Mitigation  Ensure credentials are secure. Use SSH public/private key exchange.

B.38 T3-SG-WSP: SOA Gateway Web Service Provider

B.38.1 Entry points

i. Intended Entry Point: SOAP Web services with TAS3 Security

ii. Internal Entry point: IdP to get user authentication.

iii. Internal Entry point: PDP to verify authorization
B.38.2 Asset: Web services hosted by T3-SG- WSP component

Threat 1 Read/update using SOAP Web services

**Attack** Use the existing web services to read or update existing sensitive data.

**Implemented Mitigation** Ensure TAS3 security is always enabled on the SOA Gateway.

**Implemented Mitigation** Ensure that requests that do not pass the validation are rejected, and the access attempt logged.

Threat 2 Request/Response Modification

**Attack** Parameter Tampering, Man in the Middle

**Implemented Mitigation** Request/response encoding using ZXID.

Threat 3 Unauthorized Access to the SOA Gateway services

**Attack** Read/update data using SOA Gateway services

**Implemented Mitigation** Reject requests not validated by ZXID AZ API component.

**Implemented Mitigation** Enforce authorization based on the PDP response.

B.39 T3-SP-CVT: Europass CV Transcoding Web Service

B.39.1 Entry Points

i. Intended entry point: SOAP Back office Web services without TAS3 Security

ii. Internal entry point: Authentication key (password) to secure the requesting service.

B.39.2 Assets

i. Backend temporary personal data storage (optional, used for some transformation which needs to send back a link to a file instead of the CV data itself)

ii. Service/server configuration files

B.39.3 Threats

i. Denial of service

ii. Access to sensitive data in storage

iii. XML injection using Web services, XML

B.39.3.1 Attack

Use the existing web services for:

i. XML Injection Attack

ii. Buffer Overflow Attack.

**Not existing Mitigation** Validate incoming XML using the corresponding XML Schemas (valid for XML transformation based on Europass Cedefop XML schemas, Hr-XML Schemas, IMS ePortfolio (NL) schemas but not for Leap2a or hResume (i.e. Linkedin) transformations.
B.39.3.2 Attack

Use the existing web services for:

iii. Denial of Service Attack

Not existing Mitigation  Be sure that optional support of requesting service ID is enabled (which provide a first basic level of security). Update the internal code to replace service ID support by a more secure service authentication token.

B.40 T3-SP-MATCHER: Job Profile Matching Service

B.40.1 Entry points

i. Intended Entry point: Web service interface for invoking service

B.40.2 Asset: Program logic that matches profiles of individuals to opportunities

Threat 1 Unauthorised invocation

Attack: Unauthorised call to WS interface

Not Implemented Mitigation  In order to call the matcher a valid token will be needed. The token issued will come from the main TAS3 authorisation infrastructure and present the correct credentials to the service.

Threat 2 Overload of service creating failure in audit notifications

Attack  Denial of service

Implemented Mitigation  To prevent a denial of service attack calling services will be limited to specific to those which have been pre-authenticated by the TAS3 infrastructure. Specific terms of behaviour

B.41 T3-STACK and T3-SSO-ZXID

B.41.1 Entry points

i. Shell or root shell (or ssh) administrative login

ii. TAS3 designed management interfaces (none yet)

iii. Product specific management interfaces

   - Auto-CoT (fully automatic metadata exchange and trust establishment with anonymous third party SPs)

iv. Web GUI

v. SOAP web service

   - SLO

   - WSP
B.41.2 Data assets

i. Private keys of the service itself

ii. Circle-of-Trust database

iii. Service specific user data stores

iv. Session cache, including EPRs

B.41.3 Nonfunctional assets

i. Privacy preserving through avoidance of correlation handles

ii. User consent and control of data release

iii. Organizational control of data release

iv. User choice of WSP

v. Nonrepudiation

vi. Accountability

vii. Credible authentication of system entities

B.41.4 Attacks and mitigation

• Too numerous to describe exhaustively in one afternoon *** TBD

• Generally the data assets are protected using Unix filesystem permissions against shell and local
  Unix process access. This, of course, is of little value against root. Therefore deployment MUST
  use nonroot users for running all TAS3 related processes as well as for most administrative tasks.

• The TAS3 designed and product specific management interfaces follow good coding practises (e.g.
  check for ".." in path) to only allow designed access to the data assets.

• Web GUI is coded such that only authorized accesses are possible

• SOAP web service is coded such that only authorized accesses are possible

• Appropriate crypto layer (such as TLS) is applied in Web GUI, SOAP, and ssh entry points

B.42 T3-TPN-TB2: Trust Negotiation Module

B.42.1 Entry points

Credentials and Policy Negotiation (CPN) is a web service that can be called upon by any authorised
TAS3 infrastructure component.

B.42.2 Assets

The user’s attribute credentials and release policies.

B.42.3 Threats

Exposure of the user’s attribute credentials and release policies without the user’s consent.
B.42.4 Attacks

The user’s attribute credentials are not exposed without his consent because the CPN agent enforces the user’s release policies. Preventing unauthorised exposure is the very essence of the CPN agent functionality.

However, administrative action at the CPN server’s site can also lead to unauthorised exposure. This is not prevented by technical means but should be ensured at the CPN agent’s site by implementation of physical security coupled with appropriate personnel processes.

B.43 T3-TRU-CTM: Credential based trust service Component

B.43.1 Entry points

i. Internal Entry point: Shell or root shell

ii. Intended Entry point: CTM web service Interfaces

iii. Intended Entry point: Credentials Repository Interfaces

B.43.2 Asset: Credentials

Threat 1 Credentials Disclosure/Modification.

Credentials are sensitive information encapsulating roles and attributes referring to the users. For this reason their disclosure or modification is a threat for the system.

Attack 1 Unauthorized access to the credentials repository

A malicious access to the credentials repository performed by the administrator or an unauthorised user in order to disclose or modify the stored credentials.

Existing Mitigation The use of external monitoring and logging mechanisms to trace and detect suspect activities. These mechanisms should be external to avoid the manipulation of the records performed by malicious administrators.

Implemented Mitigation The usage of an authentication mechanism to prevent the access to the credentials repository performed by unknown or unauthorized agents.

Attack 2 Man in the middle, tampering

A malicious agent gains the access to the credentials intercepting the message exchanged during the querying/updating process involving the credentials repository.

Implemented Mitigation The credentials have a digital signature that makes difficult, for a malicious agent, to fake credentials.

Existing Mitigation The use of secure channels for data transmissions (i.e. HTTPS).

Threat 2 Unauthorized Access to the CTM service functionalities

The access to the CTM service functionalities itself grants the disclosure of credentials. The response of the service, indeed, brings the credentials.

Attack Brute Force or Password Dictionary

A malicious agent gains the access to the service and obtains the credentials performing a brute force or a password dictionary attack.

Implemented Mitigation Single Sign On (SSO) authorization mechanisms to automatically and safely generate strong and secure tokens.

Existing Mitigation Mechanisms that press for the usage of strong and effective passwords.

Implemented Mitigation Logging and monitoring mechanisms to detect possible misbehaviours.
B.43.3 Asset: CTM Web Service

**Threat 1** CTM not available

A malicious agent makes the CTM web service unavailable for the intended users.

**Attack** Denial of Service (DOS) Attack

A malicious agent or a collectiveness of malicious agents saturates the CTM service with a huge amount of requests.

**Implemented Mitigation** The existence of monitoring and logging mechanisms to trace suspect behaviours.

**Implemented Mitigation** The CTM service is not registered in a public register. Only the Trust-PDP has a list of trusted CTM services.

**Existing Mitigation** The utilization of a firewall for the input output traffic filtering.

**Threat 2** Request/Response Modification

A malicious agent gains the access to the request/response messages exchanged during a CTM service call process in order to modify the results of the service.

**Attack** Parameter Tampering, Man in the Middle

The request/response messages are modified eavesdropping the network used to exchange the messages.

**Implemented Mitigation** The credentials have a digital signature that makes difficult, for a malicious agent, to fake credentials.

**Existing Mitigation** Use of secure channels for data transmissions (i.e. HTTPS)

B.44 T3-TRU-KPI: Key Performance Indicators

B.44.1 Entry points

i. Internal Entry point: KPI Data Base

ii. Intended Entry Point: Online sources of KPIs

B.44.2 Asset: KPI Engine

**Threat 1** Fake sources

**Attack** a fake host is simulating an online KPI source in order to provide garbage and fake data to the KPI engine

**Existing Mitigation** Authentication mechanism

**Threat 2** KPI values modification

**Attack** man in the middle attack, interception and modification of the KPI messages sent by the KPI sources to the KPI engine

**Existing Mitigation** Securing the communication channel (SSL, TLS)

**Existing Mitigation** Signing the content of the messages

**Threat 3** Privacy leaks

**Attack** Sniffing the traffic in order to guess the business objectives of a user or a company.

**Existing Mitigation** Securing the communication channel (SSL, TLS)
B.45 T3-TRU-RTM: Reputation Trust Management Service

B.45.1 Entry points

i. Intended Entry point: Shell or root shell
ii. Intended Entry point: RTM web service Interfaces for Trust PPD
iii. Intended Entry point: RTM web service Interfaces for the Feedback Service
iv. Intended Entry point: RTM web service Interfaces for feedback submitting
v. Internal Entry point: Feedback Repository Interfaces

B.45.2 Asset: Feedback

Threat Feedback Disclosure/Modification

Feedbacks are sensitive information because a user would like to feel free to release negative feedback without the counterpart knew about that. For these reason the disclosure or the modification of feedback represents a threat for the system.

Attack Unauthorized Access to the feedback repository

A malicious access to the feedback repository performed by the administrator or an unauthorized user in order to disclose or modify the stored feedback.

Implemented Mitigation The usage of an authentication mechanism to prevent the access to the feedback repository performed by unknown or unauthorized agents.

Existing Mitigation The use of external monitoring and logging mechanisms to trace and detect suspect activities. These mechanisms should be external to avoid the manipulation of feedback performed by malicious administrators.

Attack Man in the Middle, tampering

A malicious agent gains the access to the feedback intercepting the message exchanged during the querying/updating process involving the feedback repository.

Existing Mitigation Use of secure channels for data transmissions (i.e. HTTPS)

B.45.3 Asset: RTM Web Service

Threat 1 RTM not available

A malicious agent makes the RTM web service unavailable for the intended users.

Attack Denial of Service (DOS) Attack

A malicious agent or a collectiveness of malicious agents saturates the RTM service with a huge amount of requests.

Implemented Mitigation The existence of monitoring and logging mechanisms to trace suspect behaviours.

Implemented Mitigation The RTM service is not registered in a public register. Only the Trust-PDP has a list of trusted RTM services.

Existing Mitigation The utilization of a firewall for the input output traffic filtering.

Threat 2 Unauthorized Access to the RTM functionalities

The access to the RTM service functionalities can reveal the reputation of an agent.
**Attack**  Brute Force or Password Dictionary

A malicious agent gains the access to the service and obtains the credentials performing a brute force or a password dictionary attack.

**Implemented Mitigation** Single Sign On (SSO) authorization mechanisms to automatically and safely generate strong and secure tokens.

**Existing Mitigation** Mechanisms that press for the usage of strong and effective passwords.

**Implemented Mitigation** Logging and monitoring mechanisms to detect possible misbehaviours.

---

**Threat 3** Request / Response Modification

A malicious agent modifies the request or the response changing the reputation values for malevolent purposes.

**Attack**  Parameter Tampering, Man in the Middle

The request/response messages are modified eavesdropping the network used to exchange the messages.

**Existing Mitigation** Use of secure channels for data transmissions (i.e. HTTPS)
Annex C: User Centricity of TAS\textsuperscript{3}

C.1 Executive Summary

This document provides a first iteration of the defining elements of user-centricity in TAS\textsuperscript{3}. These elements have been identified based on existing deliverables, email discussions and meeting interaction. The subsequent sections list these elements and provide references to the deliverables that cover these aspects of user-centricity. This list will be continuously refined throughout further development of the project.

C.2 Defining elements of user-centricity in TAS\textsuperscript{3}

C.2.1 The user’s ability to express privacy preferences

Within TAS\textsuperscript{3} every data subject user will be provided with the opportunity to express his or her own privacy preferences with regards to at least the following aspects of the processing operations that take place within the TAS\textsuperscript{3} network:

- the categories of recipients of his personal data. The interface provided to the user shall be sufficiently granular to allow him to both identify categories of recipients, and also to exclude particular entities as potential recipients (e.g. to deny a particular physician future access to his/her PHR);
- what their processing capabilities shall be (e.g. read, write, edit, delete, ...) towards which data elements;
- for which context/purpose (e.g. pursuant to self-initiated job application but not for headhunting purposes; or when being referred to this doctor for treatment, etc);
- to formulate constraints (e.g. specify the time-period in which the processing operation is allowed to take place);
- whether or not an operation is to be dependent on specific obligations (e.g. notification to audit and oversight committee).

The user’s privacy preferences will be translated operationally within the TAS\textsuperscript{3} network in mainly three ways:

1. Either through a constrained delegation process (see deliverables D2.1, D7.1, D3.1);
2. Under a policy-based approach (e.g. "policy wizard") (see deliverable D4.2);
3. Or a combination of both 1 and 2.

In each of these instances the interface for the user will be the so-called "dashboard" (see D2.1), may be under control of a business process (see D3.1). Under all three approaches, the user’s privacy preferences will be translated into so-called "sticky policies", which shall be attached to the data to ensure that all data recipients along the value chain are aware of usage restrictions (and to ensure that they are subsequently enforced at every stage; and passed on further if necessary).

In order to ensure proper enforcement, the consent by the data subject shall operate as a default requirement (policy condition) for any authorization decision by Policy Decision Points (PDPs) whenever appropriate. Other consent directives (e.g. restrictions with regards to subsequent use) shall be enforced by securely associating these instructions with the data as sticky policies.

Note 1: The user’s expression of privacy preferences of course needs to take place within certain parameters. These parameters shall be clearly described in the initial privacy notice provided to the data subject.
during the intake/enrolment process. In particular, the user shall be notified of those aspects that he 
MUST subscribe to, such as processing operations, which may take place pursuant to legal obligations 
incumbent upon the user or the TAS³ network, or further processing for statistical purposes. The certain 
parameters influencing user's privacy preferences will change during participation in the TAS³ network 
dependent on the concrete process the user is involved in. E.g., the parameters are dependent on the pur-
pose and the type of usage of the user’s data, but is known at the starting of the process to some degree 
(e.g. it is not always clear which concrete service providers will be called) and is concretized during the 
process execution. This will allow to predetermine parts of the privacy preferences presenting the user the 
known information (of the business process policy) and to concretize it during execution eventually (if 
wished by the user) by additional user interaction.

Note 2: For the employability scenario there may be restrictions as to when consent may act as a legal 
basis to legitimate the processing (due to imbalance of power between employee - employer / prospective 
employee). The relevant opinions of the Article 29 Working Party will be taken into account to ensure that 
consent only acts as a legitimate basis within the meaning of articles 7 et seq. of the Directive when the 
data subject can truly give his consent "freely" (article 2 (h) of the Directive).

While pre-authorization is a possibility for relatively simple processes, more complex processes may 
require additional consent capture. After all, the user’s general privacy preferences are intended to serve 
multiple purposes, and therefore cannot adapt to all situations or remove the need for additional consent 
in case of new or unanticipated uses of information (e.g. new service, new function involving different 
recipients). In order to accommodate the need for subsequent consent capture, a "call-back" process shall 
be in place that alerts the individual to an unanticipated situation or "out-of-policy" request for use of or 
availability to information (see D2.1). In other words, for most processes the user will exercise control prior 
to the moment that a service provider requests to undertake processing (pre-authorization), for others they 
will have to authorize the transaction at the moment that it is requested (user call-back). If the authorization 
for the transaction is not possible, e.g. in the specific context the user will not give his consent, process 
adaptation (see Deliverable D3.1) can allow to change the context so that an authorization will then be 
possible, e.g., change the process so that not access to the whole data is required but only to the aggregated 
data.

C.2.2 The user’s ability to manage his own partial identities

In addition to deciding which attributes he discloses to which service provider (and under which condi-
tions), the user will also have the opportunity to choose which digital identity (identity provider or other 
authoritative source for attribute information) he uses to provide these attributes.

In this regard the TAS³ approach is somewhat similar to the Microsoft Cardspace model; however, 
the TAS³ approach is more advanced for two main reasons. First, the user has the ability to become 
actively involved in the management of the identifiers/ pseudonyms associated with his respective 
digital identities, and the correlations between them. Second, the TAS³ approach provides for an impor-
tant functionality currently not provided by Cardspace, namely the ability to aggregate attributes across 
different partial identities to respond to a single request from a service provider, without compromising the 
privacy of the data subject with regards to the identifiers associated with these different partial identities.

Another advantage provided by TAS³ is the governance framework. The contract framework, coupled 
with the required polices, creates an ecosystem-wide binding of obligations. Most systems can only bind 
a limited number parties to a transaction and then only for a limited number of transactions.

See deliverables D2.1 (high-level), D4.2 D7.1 and upcoming deliverables of WP6.

C.2.3 The user’s ability to express trust preferences and provide feedback

The user’s ability to express trust preferences in TAS³ is accommodated by allowing the user to specify 
the "trust rating" or "trust score" that is required for entities in order for them to be involved in processing 
operations involving his personal data. If the trust ranking of a chosen entity falls below the level specified 
by the user before the process is completed, the user will be notified and given the opportunity to terminate 
the process.
Example: A user may specify that head-hunters are authorized to access his e-Portfolio for placement purposes, but he trusts only head-hunters with a sufficiently high trust rating. This condition then applies cumulatively along with the user’s specified privacy preferences. So head-hunter X may initially have been authorized to access the user’s e-Portfolio as far as the trust and privacy preferences were concerned (because the user has specified that his e-Portfolio may be accessed by head-hunters for placement purposes). If the trust rating of the head-hunter drops during the process (because other users give negative feedback on him), the head-hunter fails to meet the required trust rating and is denied access. The user is then notified and may decide between choosing another service provider, changing the further processing logic, or terminating the process.

For this purpose, the user is provided with a feedback mechanism, in which he can share experiences with regard to particular service providers. The resulting feedback in turn affects the overall "trust rating" of the service provider in question.

See deliverable D5.4 (expression of trust preferences into policies), D5.2 (trust management based on user feedback), D2.1 (subrole of auditor); upcoming deliverables in WP 6 will include the contract related to reputation based service providers and any oversight processes/policies to help assure correctness and fairness; upcoming deliverables in WP 3 (especially D3.3, D3.1) provide the ability to apply and dynamically adapt trust policies and feedback in a process-specific context, this is handled in policies and supported by process-controlled user interactions.

C.2.4 The user’s ability to view his personal data

The user will have the ability to view his personal data through the dashboard. As part of the trust network each data holder shall be obligated to provide a service that allows the user to view her personal data. Additionally, the dashboard will provide the data subject with an overview of Service Providers that have accessed her personal data.

C.2.5 Enhanced transparency

TAS³ shall ensure that, as a rule, no operation upon personal data will be authorized within the TAS³ network without the prior consent of the data subject.

As described in section of 3.2 of D6.2, notice and consent typically only provide ex ante transparency towards the data subject. The data subject usually has no or only limited means of verifying whether or not the data recipient has adhered to the asserted or negotiated policies.

TAS³ will enhance transparency towards the data subject by providing him with the opportunity to verify after the fact which actions upon his personal data have taken place. Due to the advanced level of security and accountability mechanisms applied throughout the TAS³ network, the user will be able to obtain a much higher degree of assurance that his privacy preferences have in fact been adhered to.

See deliverable D2.1 (dashboard)
**Bibliography**


[AcctSvc] "Liberty ID-WSF Accounting Service Specification"


[CardSpace] InfoCard protocol (aka CardSpace) from Microsoft


[DesignPat] "Liberty ID-WSF Design Patterns", liberty-idwsf-dp-v1.0.pdf


[Disco12] Liberty ID-WSF Discovery service 1.2 (liberty-idwsf-disco-svc-v1.2.pdf)

[DST11] Liberty DST v1.1


[FF12] Liberty ID Federation Framework 1.2, Protocols and Schemas


[Hardt09] Dick Hardt and Yaron Goland: "Simple Web Token (SWT)", Version 0.9.5.1, Microsoft, Nov. 4, 2009 (SWT-v0.9.5.1.pdf)


"Deliverable D6.2 RA Model V2.0", All NEXOF-RA Partners, NESSI Strategic Project and External Contributors, 2009.


http://oauth.net/

http://openid.net/


Roel Peeters, Koen Simoens, Danny De Cock, and Bart Preneel: "Cross-Context Delegation through Identity Federation", KUL 2009 (To be published?)


P. Deutch: "GZIP file format specification version 4.3", Aladdin Enterprises, May 1996

S. Bradner, ed.: "Key words for use in RFCs to Indicate Requirement Levels", Harvard University, 1997.


M. Smith: "Definition of the inetOrgPerson LDAP Object Class", Netscape Communications, RFC 2798, April 2000.

S. Josefsson, ed.: "The Base16, Base32, and Base64 Data Encodings", July 2003. (Section 4 describes Safebase64)


OASIS. "Level of Assurance Authentication Context Profiles for SAML 2.0" Working Draft 01. 01 July 2008

SAML 1.1 Core, OASIS, 2003

"Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1", Oasis Standard, 2.9.2003, oasis-sstc-saml-bindings-1.1


[SWIG] Simplified Interface and Wrapper Generator by Dave Beazley. www.swig.org


[TAS3CONSOAGMT] "TAS3 Consortium Agreement", TAS3 Consortium, 2008. (Not publicly available.)
[TAS3D12DESIGNRAR] David Chadwick (Kent), Seda Gürses (KUL), eds.: "Requirements Assessment Report", TAS3 Consortium, 20090102. Document: TAS3_D1p2_Requirements_Assessment_Report_1_V1p0.pdf


[Tom09] Allen Tom, et al.: "OAuth Web Resource Authorization Profiles (OAuth WRAP)", Version 0.9.7.2, Google, Microsoft, and Yahoo, Nov. 5, 2009 (WRAP-v0.9.7.2.pdf)
