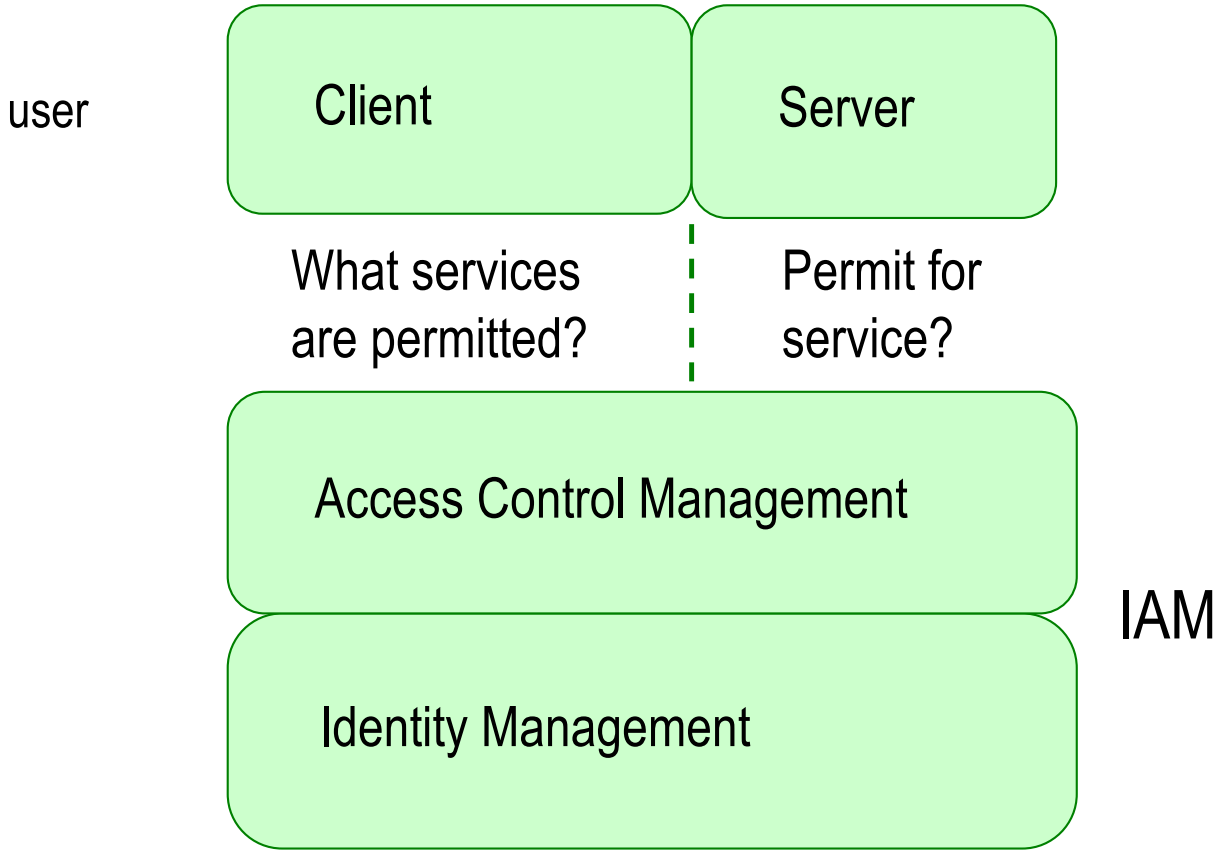
Access Control For Services

Ulf Schreier

Based on Ph.D. work of Marc Hüffmeyer

Access Control (Authorization)



Focus on general ideas, but with view on RESTful Web services

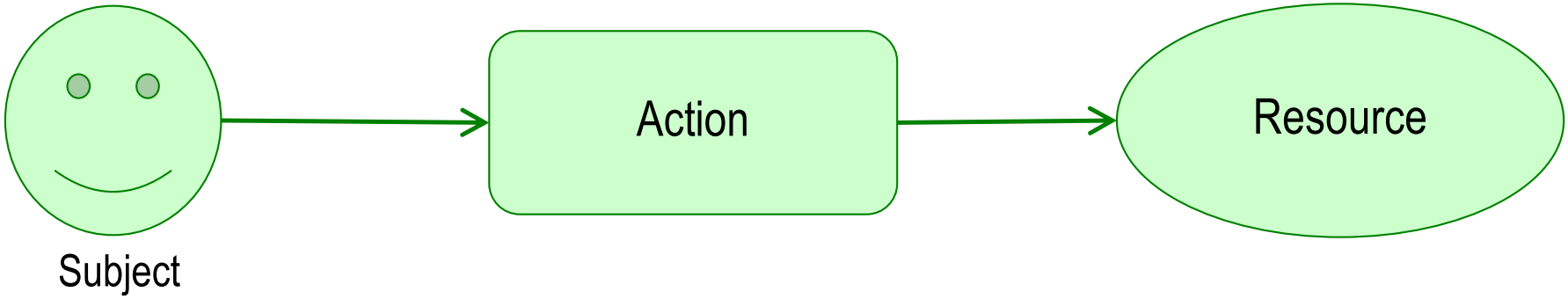
Objectives Of Talk

- Review of AAA concepts
  - Authentication
  - Authorization (Delegation)
  - Access Control (Authorization)
- How does access control get its data?
- Customization of access control towards RESTful web services

Overview

1. **Basics of access control**
2. Authentication, authorization and access control
3. ABAC for REST: RESTACL
4. (If time: IoT-oriented applications of ABAC for REST)
5. Conclusions

Basic Access Control Model



Has subject S permit to access resource R by action A?

Categories: Subject, Action, Resource

Variations of Access Control: ACL

- Access Control List (ACL)
- Structure: (Ri, Ai, {Si, ..., Sj})
- Example:

  An user 1 has permit to read heating data for sensor 1.

Variations of Access Control: RBAC

- Role Based Access Control (RBAC)
- Each subject has one or more roles RO
- Structure: (Ri, Ai, {ROi, ..., ROj})
- Example:

  A user with role „heating control service“ can read data for sensor 1.

Variations of Access Control: ABAC

- Attribute Based Access Control (ABAC)
- Structure: (conditions on R, A, S and other categories)
- Example:

  A staff member of heating control service H can read heating data from 9am to 5pm for appartment A, if there is an agreement for A with H.

subject

environment

resource

ABAC implementations

- IAM systems with XACML (OASIS standard)
- Microsoft Dynamic Access Control
- Cloud platforms with restricted forms:
  Amazon Web Services, Open Stack
- RESTACL: customized to RESTful services

- Reality for many business applications:
  - procedural programming inside transactions

Policy Administration With XACML

```
policyset {
  apply denyOverrides
  target clause URLresource == "https://smarthome.com/building"
    policy {
      apply permitOverrides
      target clause actionMethod == "GET“
        rule {
          target clause subjectName == ”Alice”
          permit
        }
    }
}
```

- ALFA syntax
- axiomatics.com
- OASIS draft

To summarize: Nested sequence of If-statements

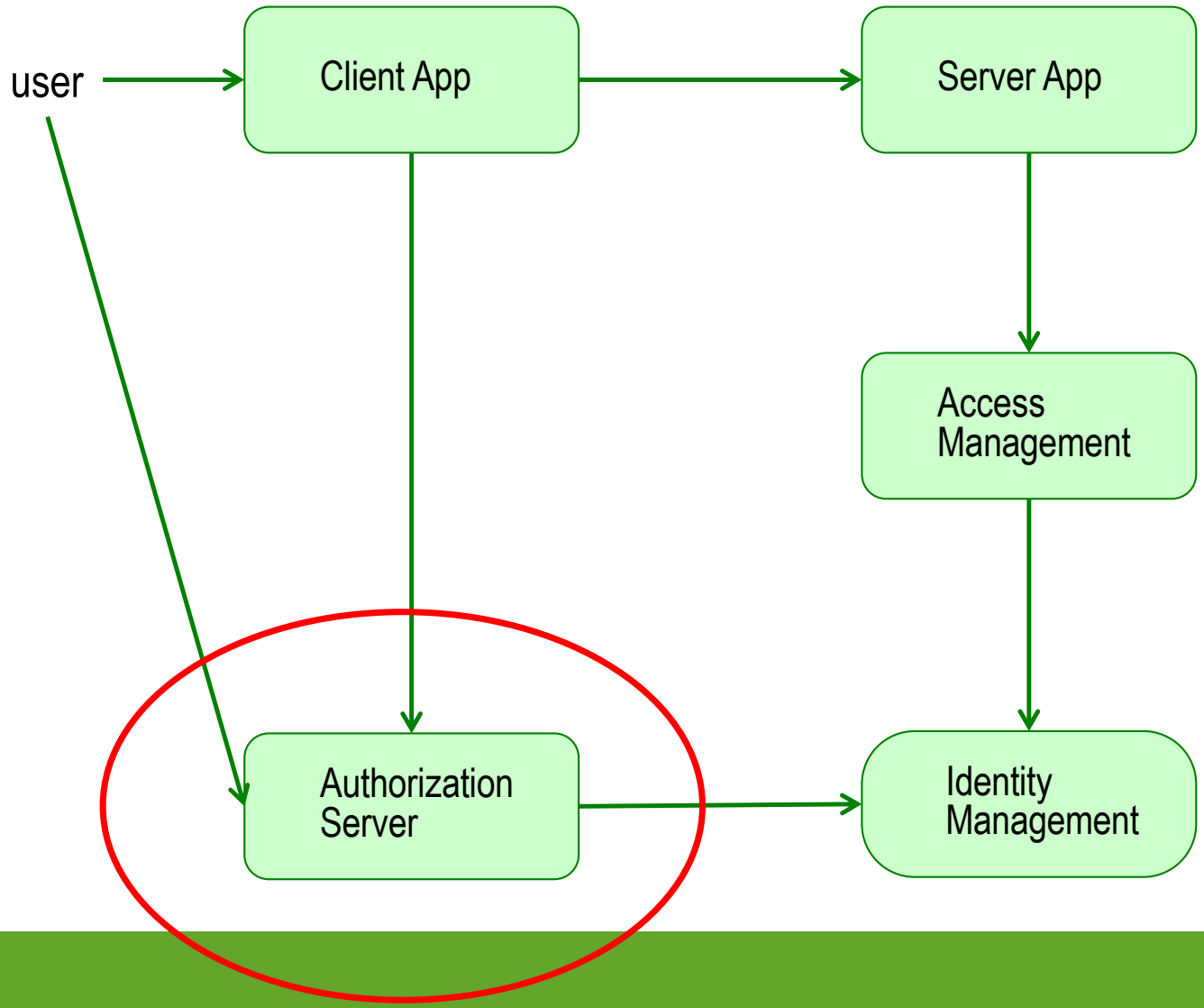
XACML Policies, Sets And Combining Strategies

- Nested policy sets
  - Controlled by target conditions
- Complex nested combining strategies
  - **Deny-override**: first deny rule evaluation decides
  - **Permit-override**: first permit decides
  - **First-applicable**: first applicable rule decides (either permit or deny)
  - **Only-one-applicable**: if not only one, result is “indeterminate“
  - Each set has its own combining

Overview

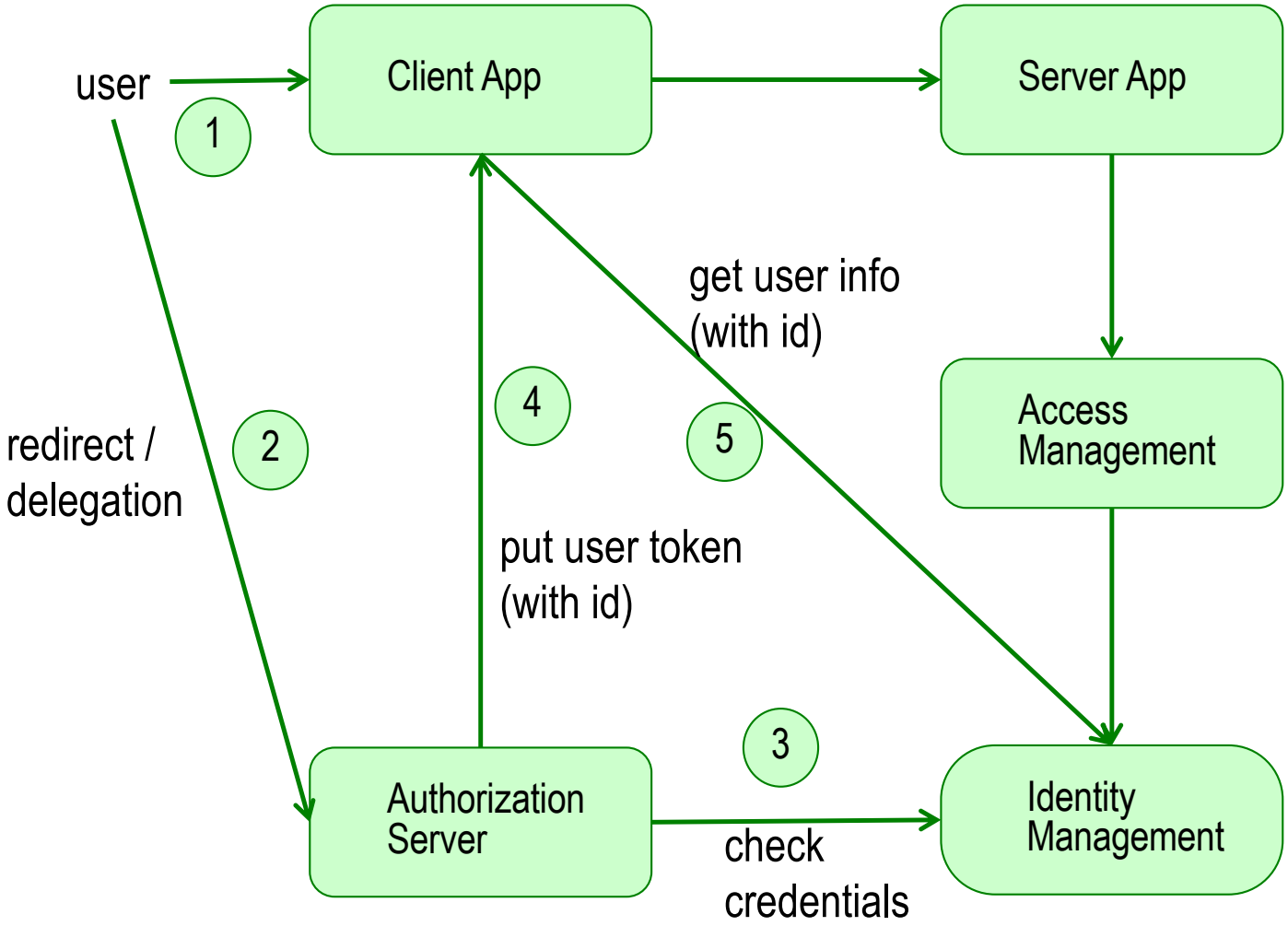
1. Basics of access control
2. **Authentication, authorization and access control**
3. ABAC for REST: RESTACL
4. Conclusions

Access Control In The Context Of Distributed Services



How does access control get its data?

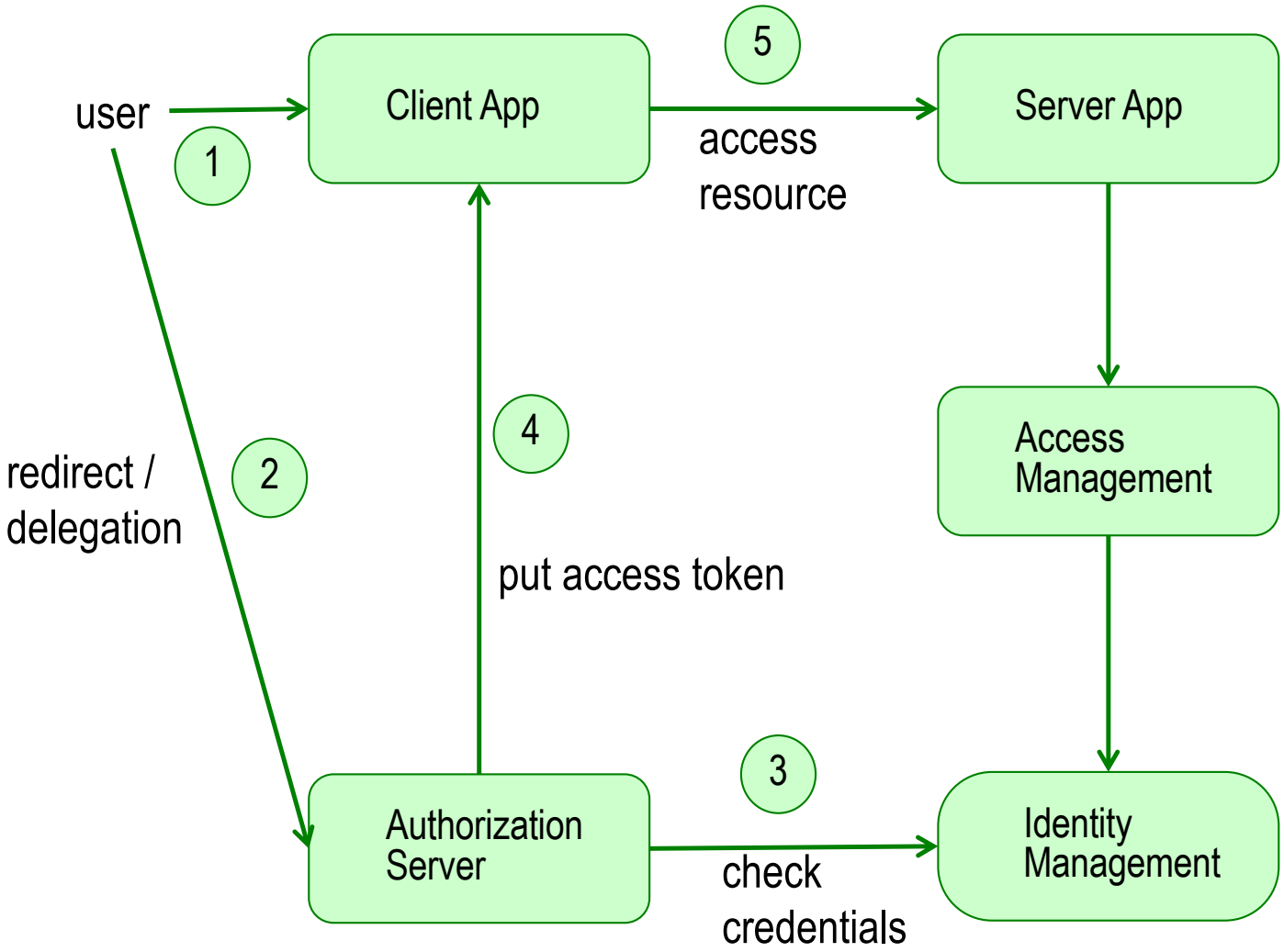
1: Authentication And User Info



Delegation of identity management as microservice

OIDC standard for REST

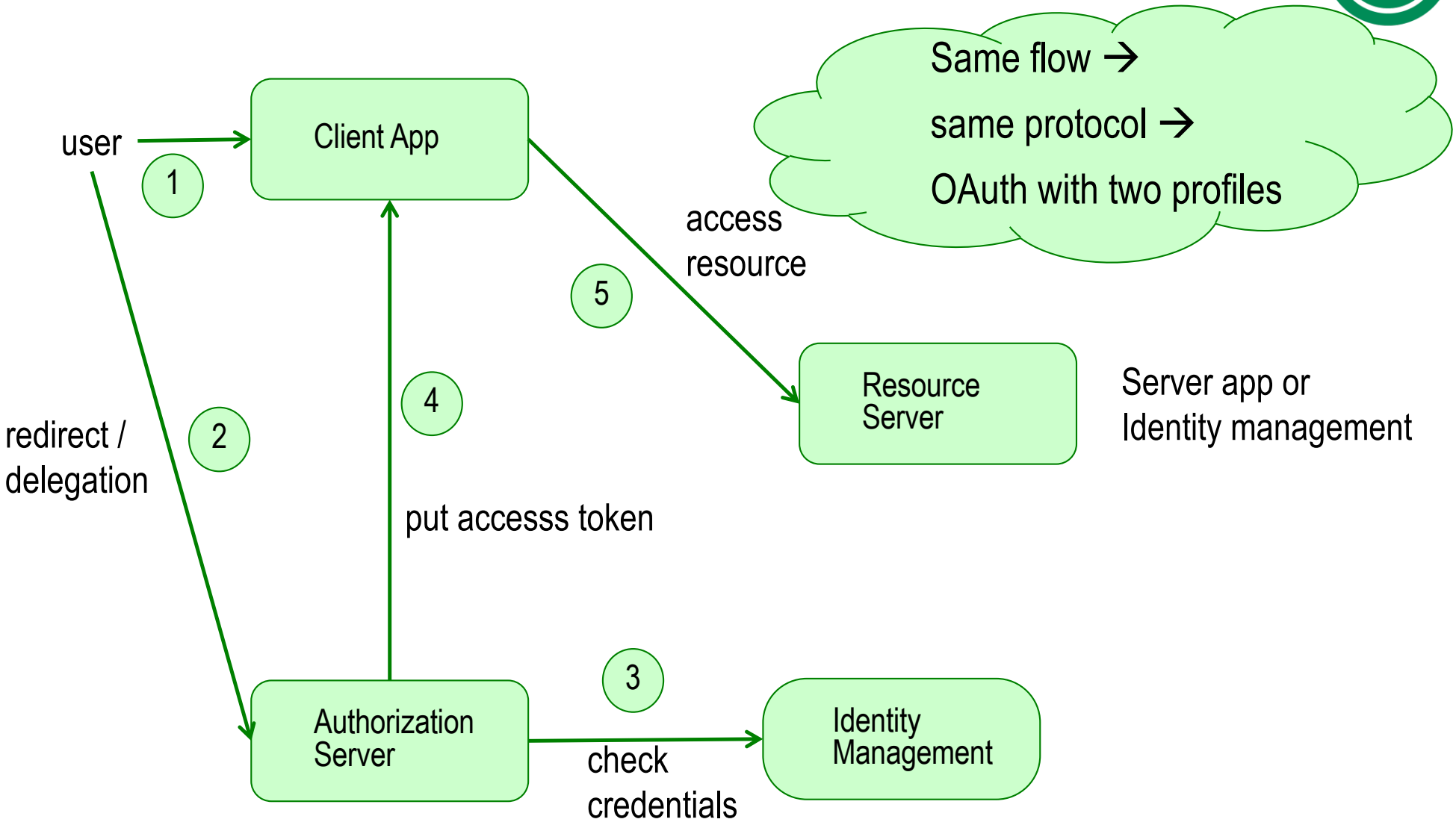
2: Delegation (Authorization)



Client access on server
on behalf of user
on **OWN DATA**
without password

OAuth for REST

Authentication and Authorization: Same flow



OAuth 2.0 Quick Facts

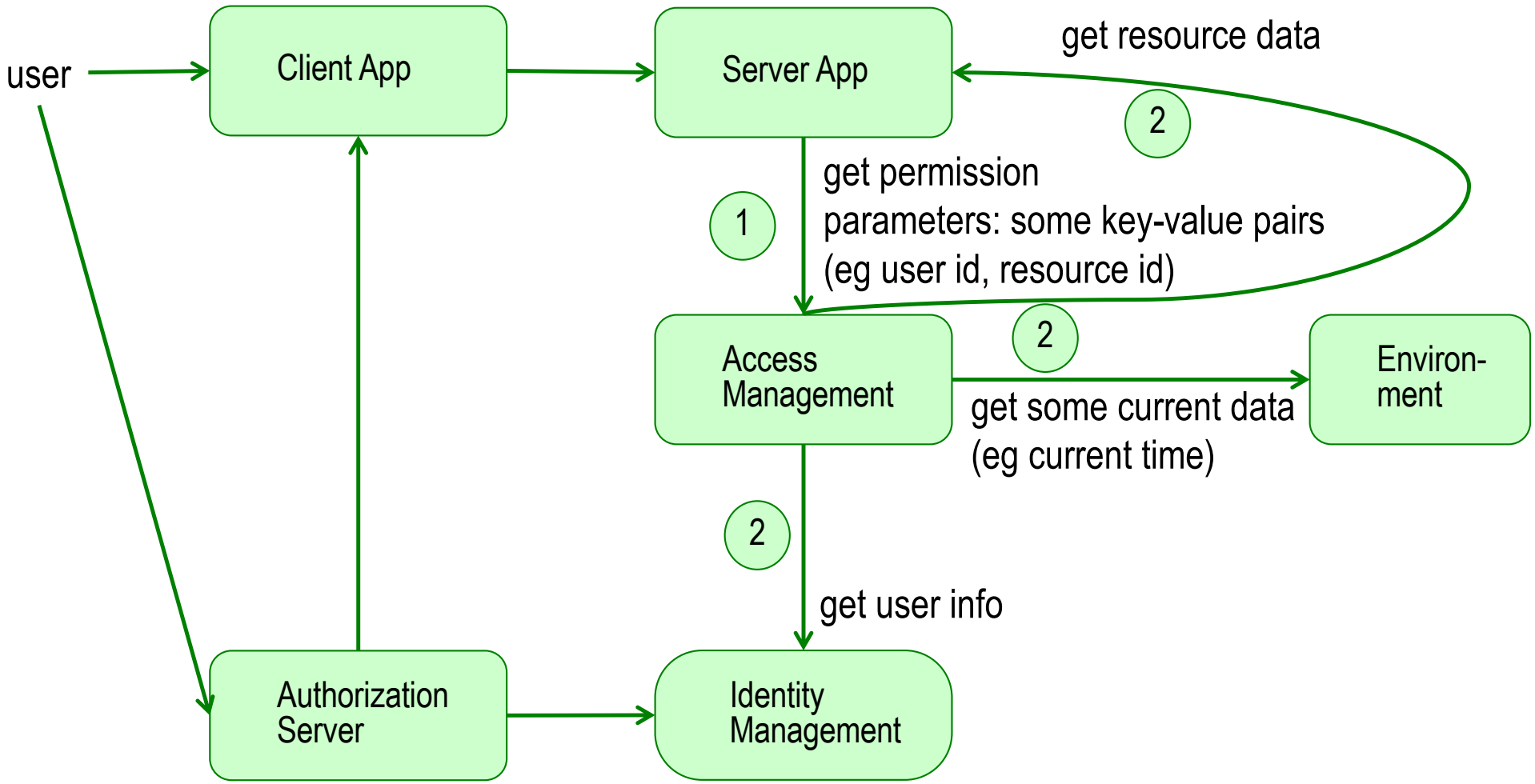
- OAuth = Open Authorization Framework
- For RESTful environments
- Security as simple as possible
- Similar ideas with
  - SAML (SOAP services)
  - Kerberos (local (Windows) networks)
- Eg. Google applications
- References: IETF standard 6749

Ticket (Token) Handling

OpenID Connect (OIDC) Quick Facts

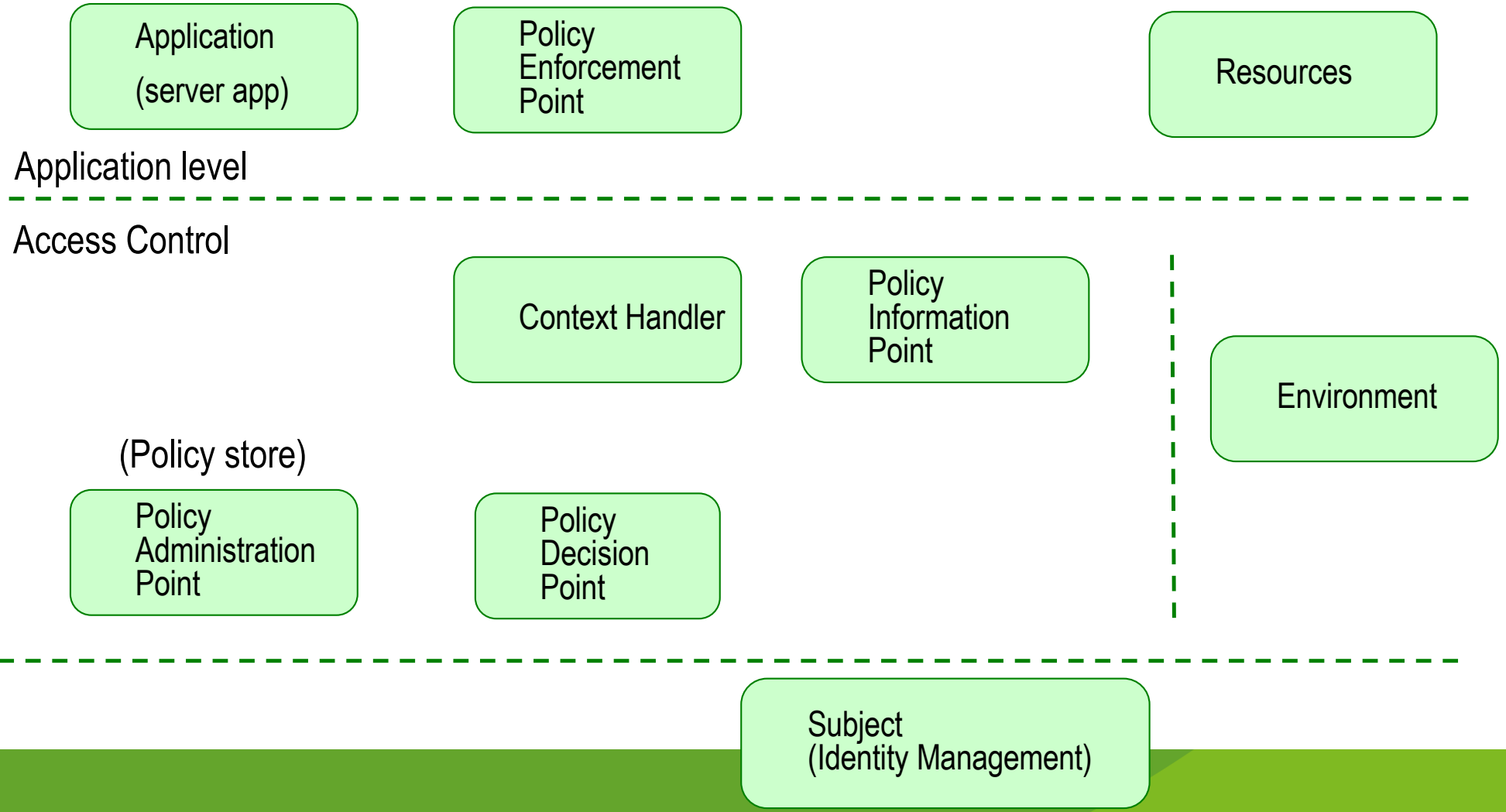
- Standard of OpenID Foundation
- Using OAuth
  - Identity management as resource server
  - Specialised token (user token with user id)
- Eg. Login with Facebook, Google, etc. account

3: Access control (Authorization)

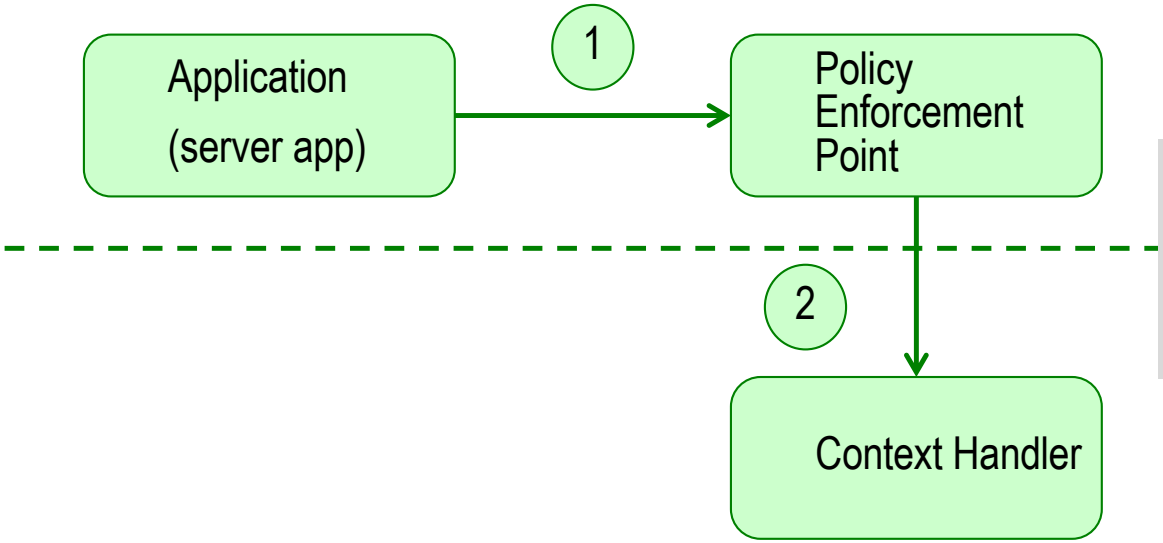


Architecture Of Attribute Based Access Control

(extended form standardized as part of XACML)



Architecture Of Access Control



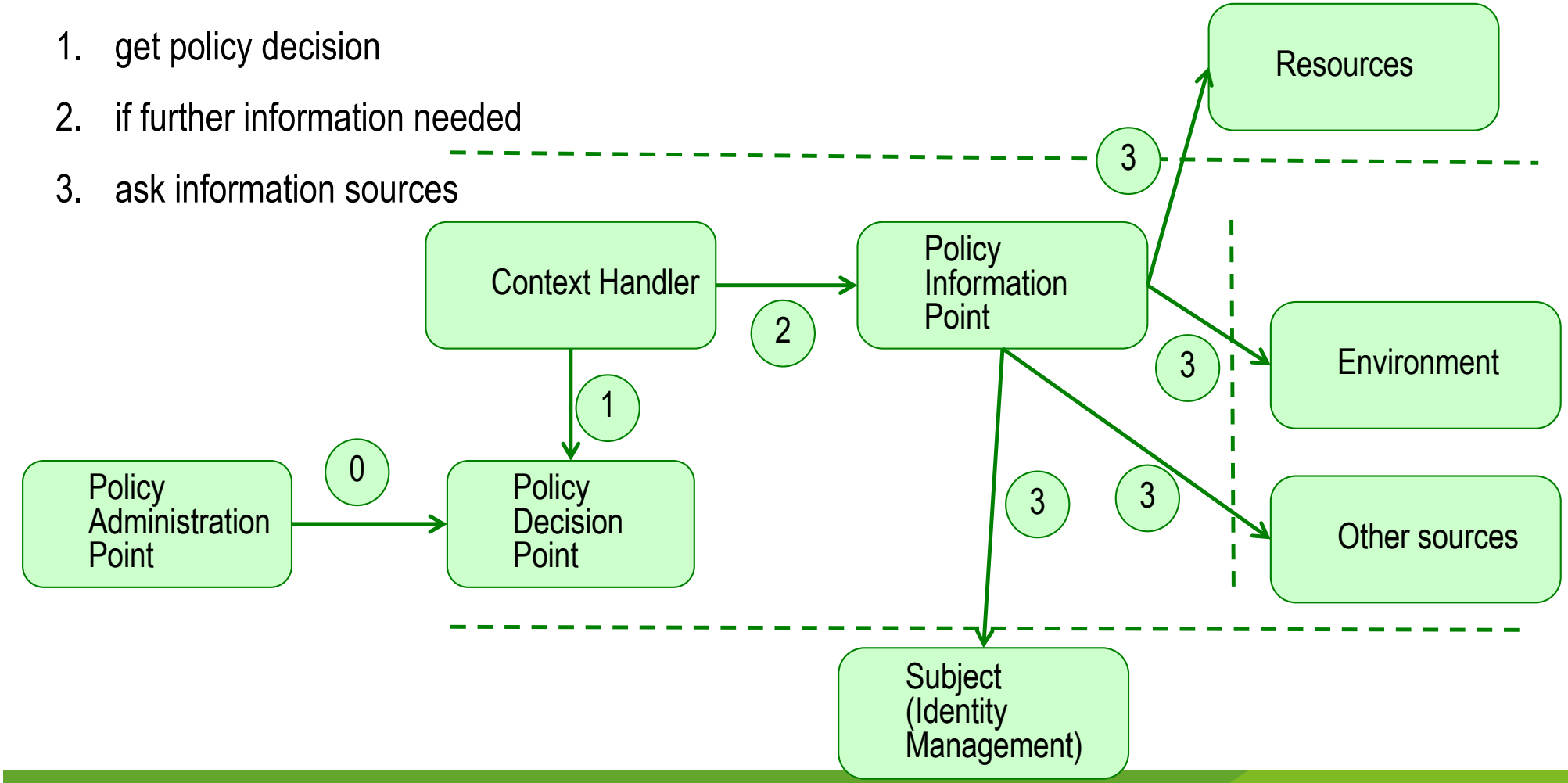
Example: subject : userid = 42
action : host = http://localhost
action : name = GET
resource : file-id = 4711

1. Application → Policy Enforcement Point (PEP)
   - no standardized interface
   - Could be
     - application logic dependent library call
     - generic filtering of http request by server

2. PEP → Context handler
   - Standardized by XACML
     - XML or SAML or JSON format
     - REST endpoint
   - (Category, Key, Value) triples

Architecture Of Access Control

0. load policies (out-of-band)
1. get policy decision
2. if further information needed
3. ask information sources



Overview

1. Basics of access control
2. Authentication, authorization and access control for services
3. **ABAC for REST: RESTACL**
4. Conclusions

Uniform Interfaces of RESTful Web Service

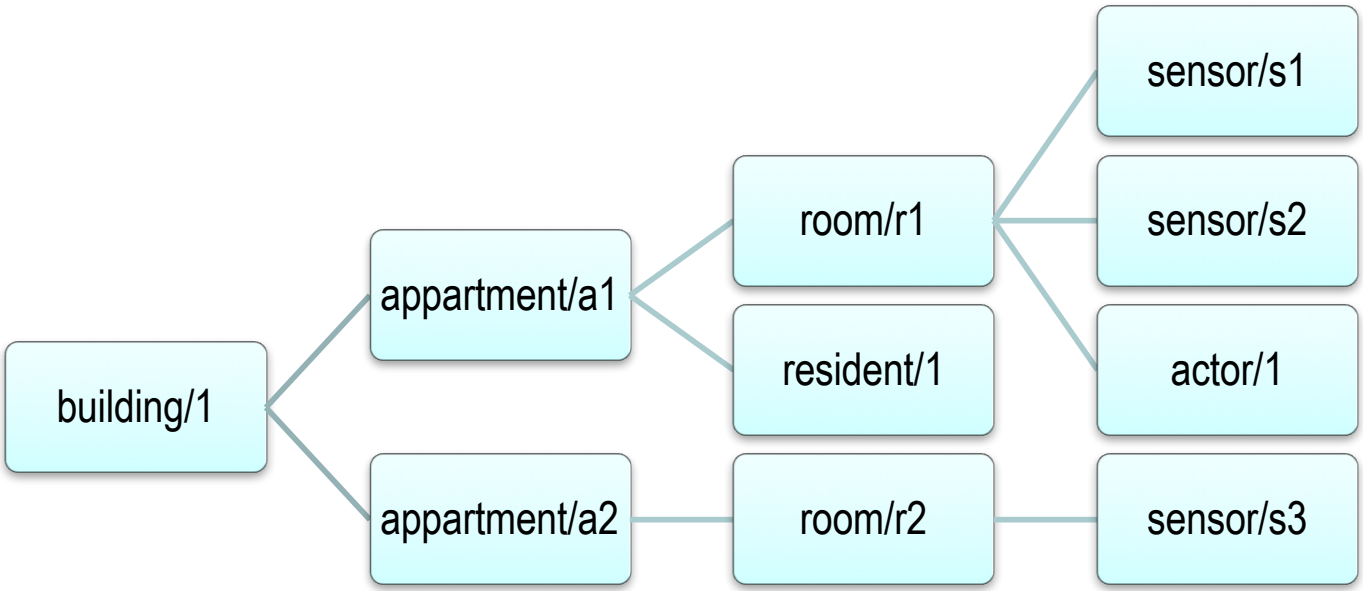
- Unique identification of resources
- Generic actions
- Manipulation of resources through representations
- Self-descriptive messages

- Mainly HTTP protocol with URL

```
GET /building/b1 HTTP/1.1
Host: smarthome.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

Sample URL Resource Trees for a Smart Home

- /building/{bid}/appartment/{aid}/room/{rid}/sensor/{sid}
- /building/{bid}/appartment/{aid}/room/{rid}/actor/{}
- /building/{bid}/appartment/{aid}/resident/{uid}



XACML For REST Resource Trees

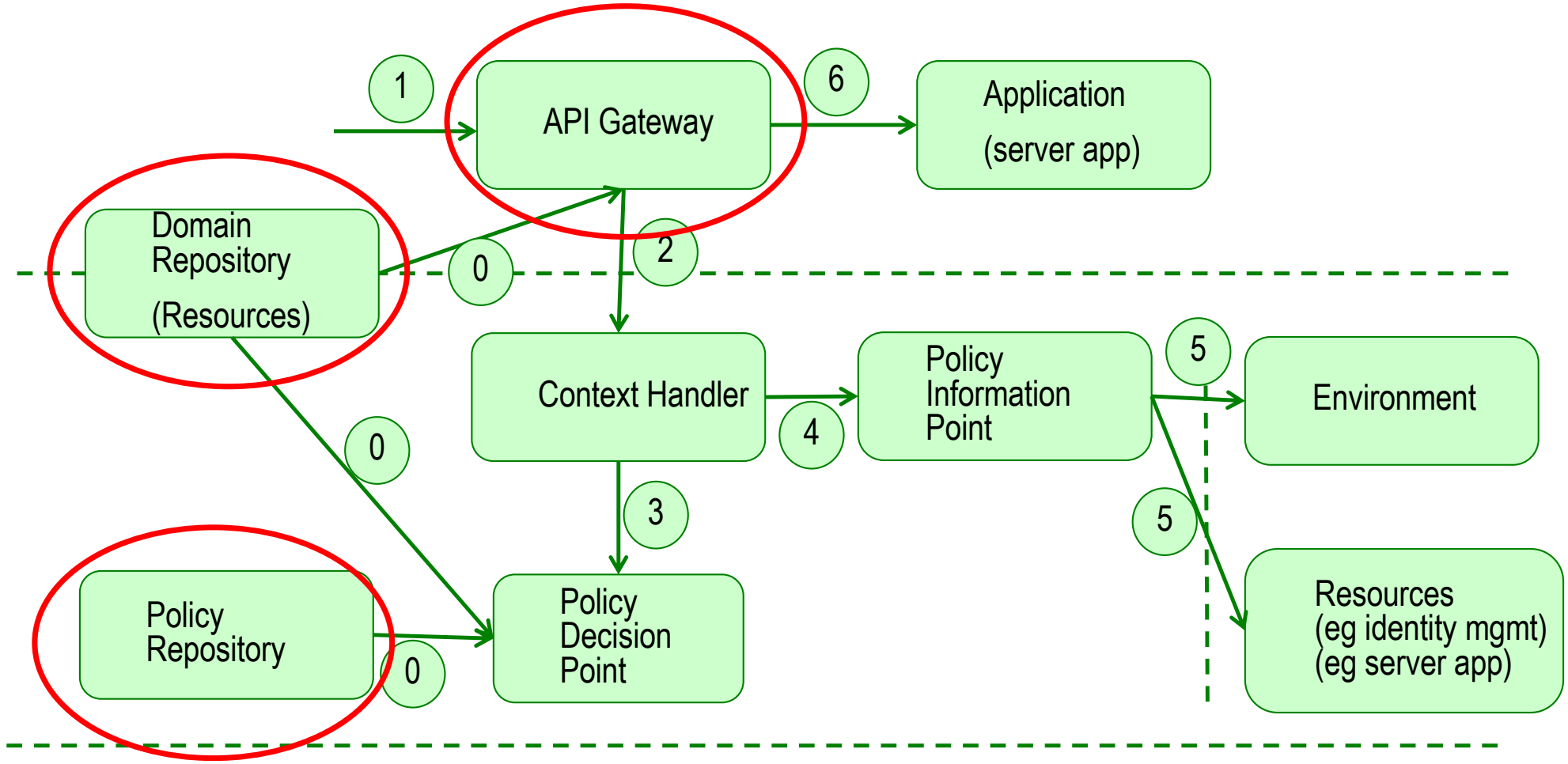
```
policyset {
  policyset {
    target clause URL == "/building/1/.*"
      policyset {
        target clause URL =="/building/1/appartment/1/.*"
        policyset {
          target source URL == "/building/1/appartment/1/room/1/.*"
            policy for sensors
        }
        …
      }
     ….
   }
   …
}
```

RestACL

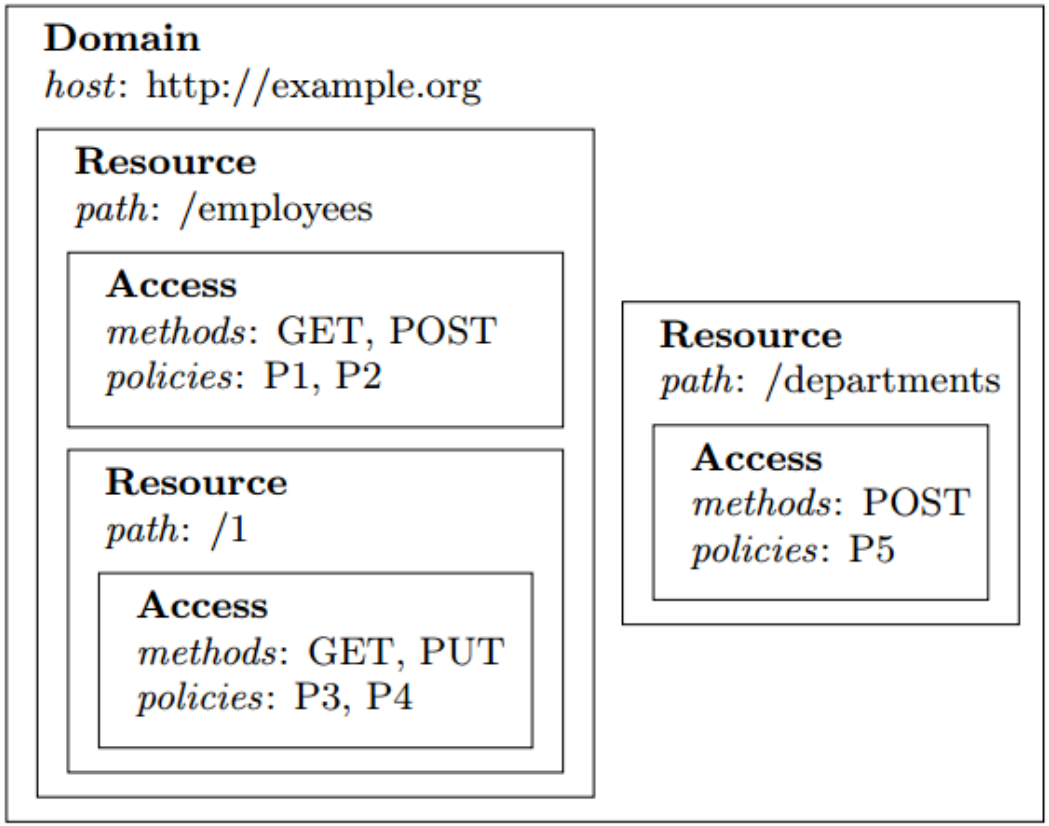
- Policy Repository
  - Collection of ABAC policies
- Domain
  - Customized for RESTful services
  - Index based structure to quickly map from requested resources to policies
- Evaluation Engine
  - Simple priority based combining strategy (but equivalent to XACML)

- References
  - Marc Hüffmeyer und Ulf Schreier, Analysis of an Access Control System for RESTful Services, ICWE'16
  - Marc Hüffmeyer und Ulf Schreier, Formal Comparison of an Attribute Based Access Control Language for RESTful Services with XACML, SACMAT'16

Architecture Of ABAC for REST (+ RESTACL)



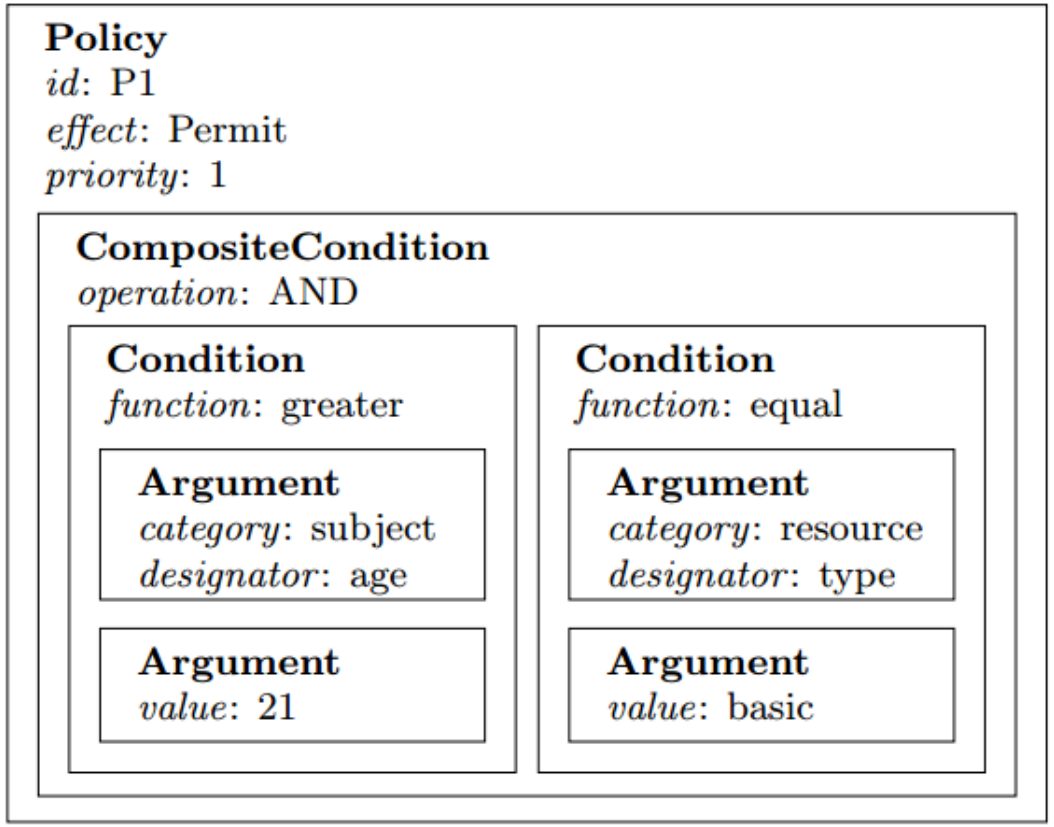
RestACL – Domain




```
{
  "host": "http://example.org",
  "resources": [{
    "path": "/employees",
    "access": [{
      "methods": ["GET, POST"],
      "policies": ["P1","P2"]
    }],
    "resources": [{
      "path": "/1",
      "access": [{
        "methods": ["GET, PUT"],
        "policies": ["P3","P4"]
      }]
    }]
  },{
    "path": "/departments",
    "access": [{
      "methods": ["POST"],
      "policies": ["P5"]
    }]
  }]
}
```

Adjusting to SWAGGER representation

RestACL – Policy




```
{
  "policies": [{
    "id": "P1",
    "effect": "Permit",
    "priority": "1",
    "compositeCondition": {
      "operation": "AND",
      "conditions": [{
        "function": "greater",
        "arguments": [{
          "category": "subject",
          "designator": "age"
        },{
          "value": "21"
        }]
      },{
        "function": "equal",
        "arguments": [{
          "category": "resource",
          "designator": "type"
        },{
          "value": "basic"
        }]
      }]
    }
  }]
}
```

Logical expressions and priorities

RestACL – Implementation

Domain: Hash table

- Hash: resource address
- Bucket: Access elements pointing to policies

Hash(Resource add.)	Access
h(/employees)	[GET, POST]
h(/employees/1)	[GET, PUT]
h(/departments)	[POST]

Policy	P1
effect	permit
priority	1
condition	…

Policy	P3
effect	deny
priority	3
condition	…

Summary on ABAC4REST/RESTACL

Complexity	Processing Time	Memory Consumption
RestACL	Constant	Growing like hashing
Intuitive XACML rule sets	Far slower	Linear growth
Nested optimized XACML policy sets	Constant (but far slower)	Growing like a data tree

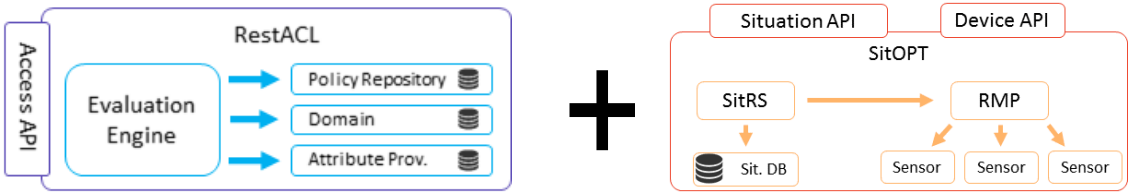
- ABAC4REST: Adaption of general access control architecture
- RESTACL:
  - Special language for REST applications
  - Fast resource-oriented implementation

Overview

1. Basics of access control
2. Authentication, authorization and access control for services
3. ABAC for REST: RESTACL
4. **(IoT-oriented applications of ABAC for REST)**
5. Conclusions

IoT Applications of ABAC4REST

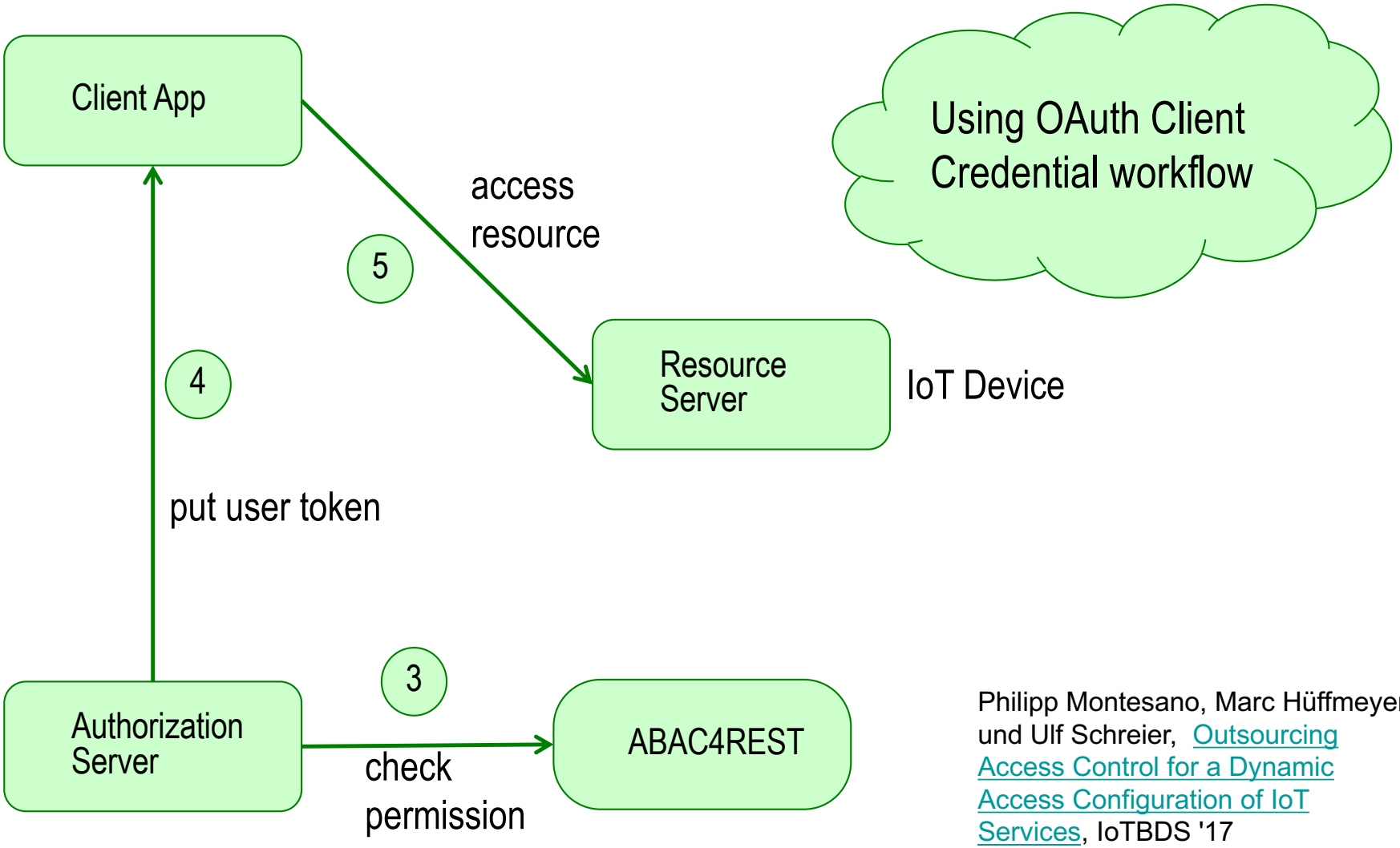
- Proof of concept with two case studies
  - Outsourcing access control from IoT
    - IoT device as OAuth resource server
    - ABAC instead of user based permissions
  - Integration with IoT situation rule system SitOPT (University Stuttgart)



Outsourcing Access Control from IoT Device

- Assumption: IoT device
  - small system
  - weak capacity
- Access control as
  - an external service
  - providing permit tickets

Outsourcing Access Control from IoT Device



Philipp Montesano, Marc Hüffmeyer und Ulf Schreier, Outsourcing Access Control for a Dynamic Access Configuration of IoT Services, IoTBDS '17

Results

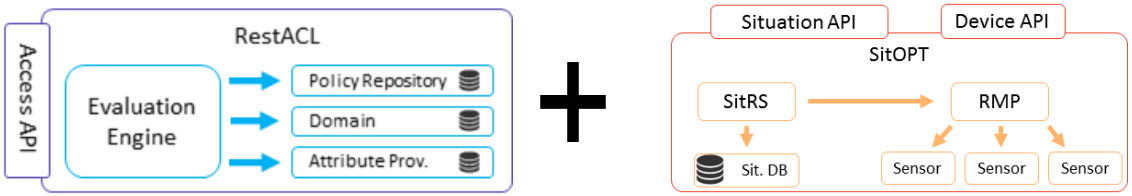
- Advantages
  - Less memory consumption at IoT device
  - Central administration of policies
- Disadvantages
  - More communication time
  - Decryption of signed tickets needs time

Situation based access control: Idea

Introduce Situation Category

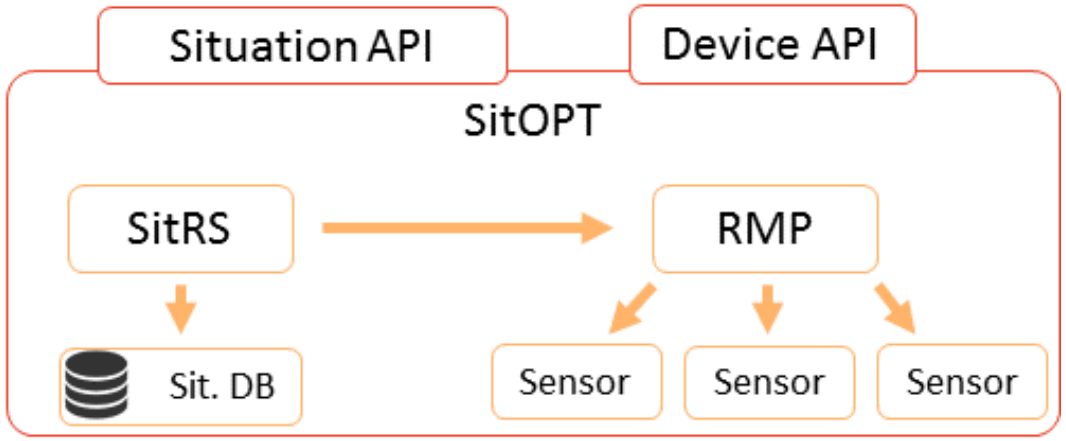
Situation
- Id
- Time
- Occurred
- AccessInterval

Integrate ABAC with Sit. Recog.



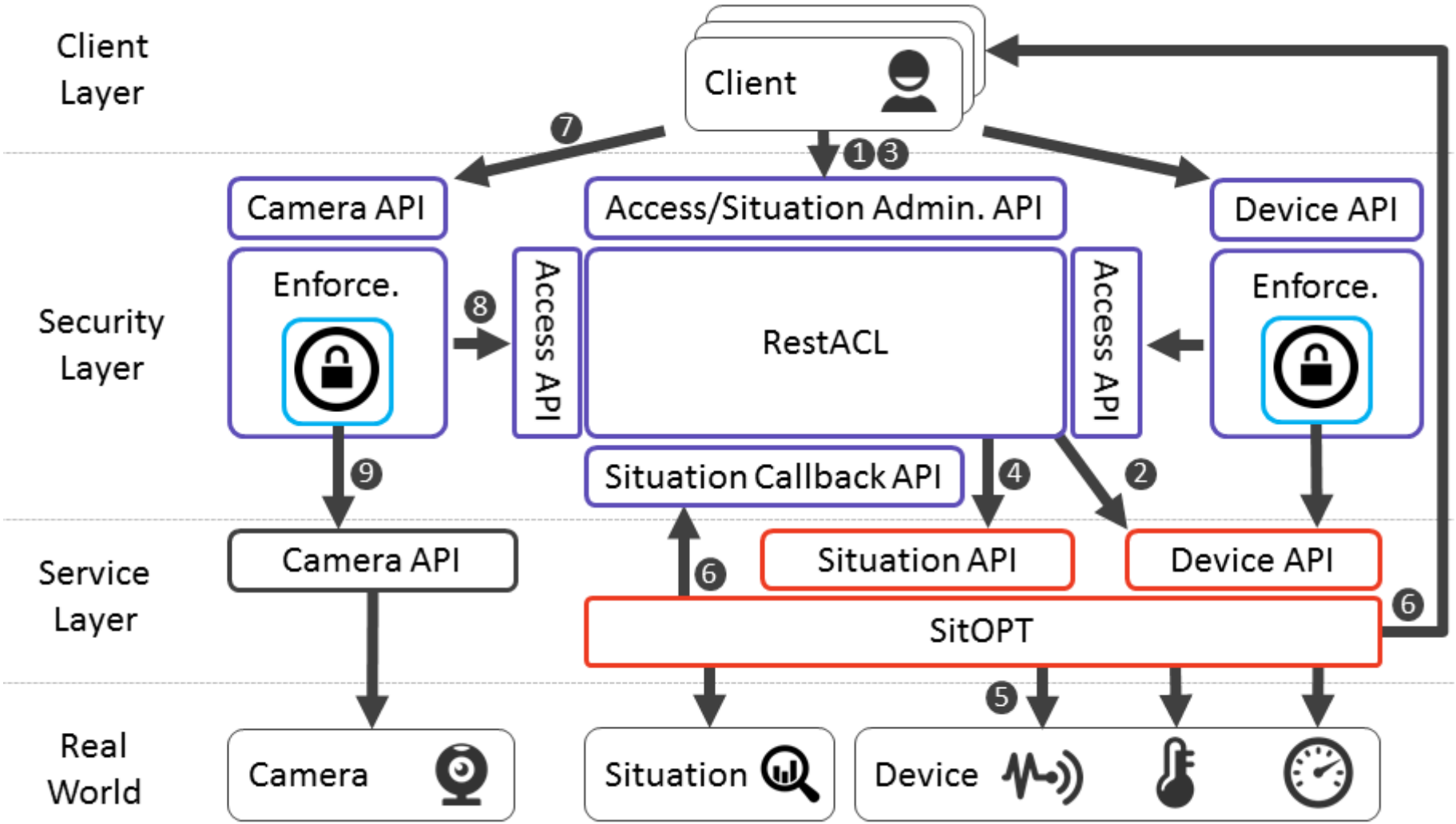
Marc Hüffmeyer, Pascal Hirmer, Bernhard Mitschang, Ulf Schreier und Matthias Wieland, SitAC - - A System for Situation-Aware Access Control - Controlling Access to Sensor Data, ICISSP '17

SitOPT (University Stuttgart)

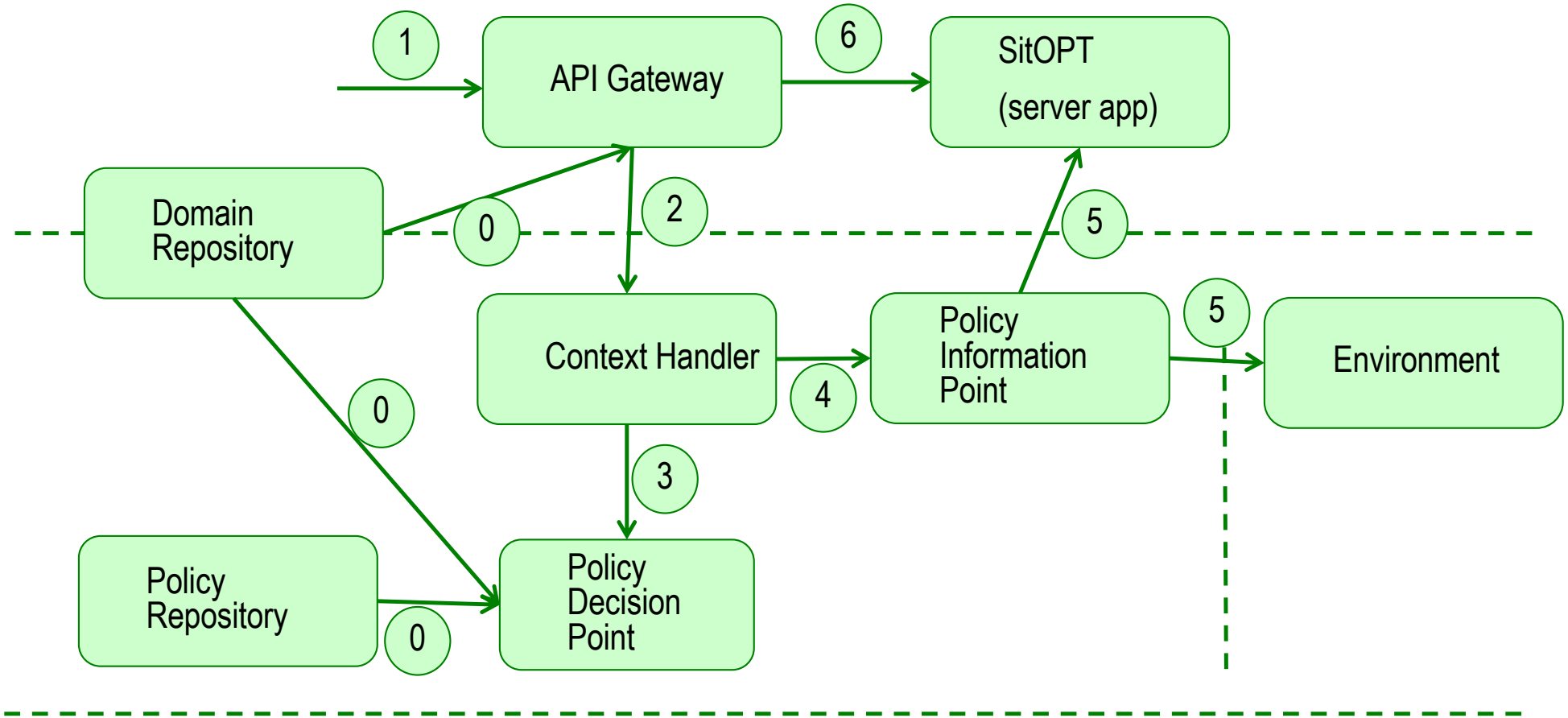


- Situation Recognition Service (SitRS)
  - Periodically checks if a situation occurred based on sensor values.
- Resource management platform (RMP)
  - Provides a uniform interface for sensors.
- Situation Database (SitDB)
  - Changes to the situation (occurred/not occurred) are written into the SitDB.
  - For every change a situation callback can be alerted.

SitAC: Architecture



ABAC4REST Architecture For SITOPT



Results

- Successful integration of two complex systems
  - Test of ABAC4REST architecture
  - Test of RESTACL language and engine

Conclusions

- Clarification of sometimes confusing and intersecting AAA concepts
  - Authentication
  - Authorization (Delegation)
  - Access Control (Authorization)

- Workflow of access control
  - Black box (access control and environment)
  - White box (inside)

- Customization towards RESTful services:
  - ABAC4REST architecture
  - RESTACL language and engine

Thank You!