CAEP Deep Dive

Implementing Session Revocation and Authorization

Atul Tulshibagwale

CTO, SGNL

Tim Cappalli

Identity Standards Architect, Microsoft

Co-chairs, Shared Signals WG

OpenID Foundation

Agenda

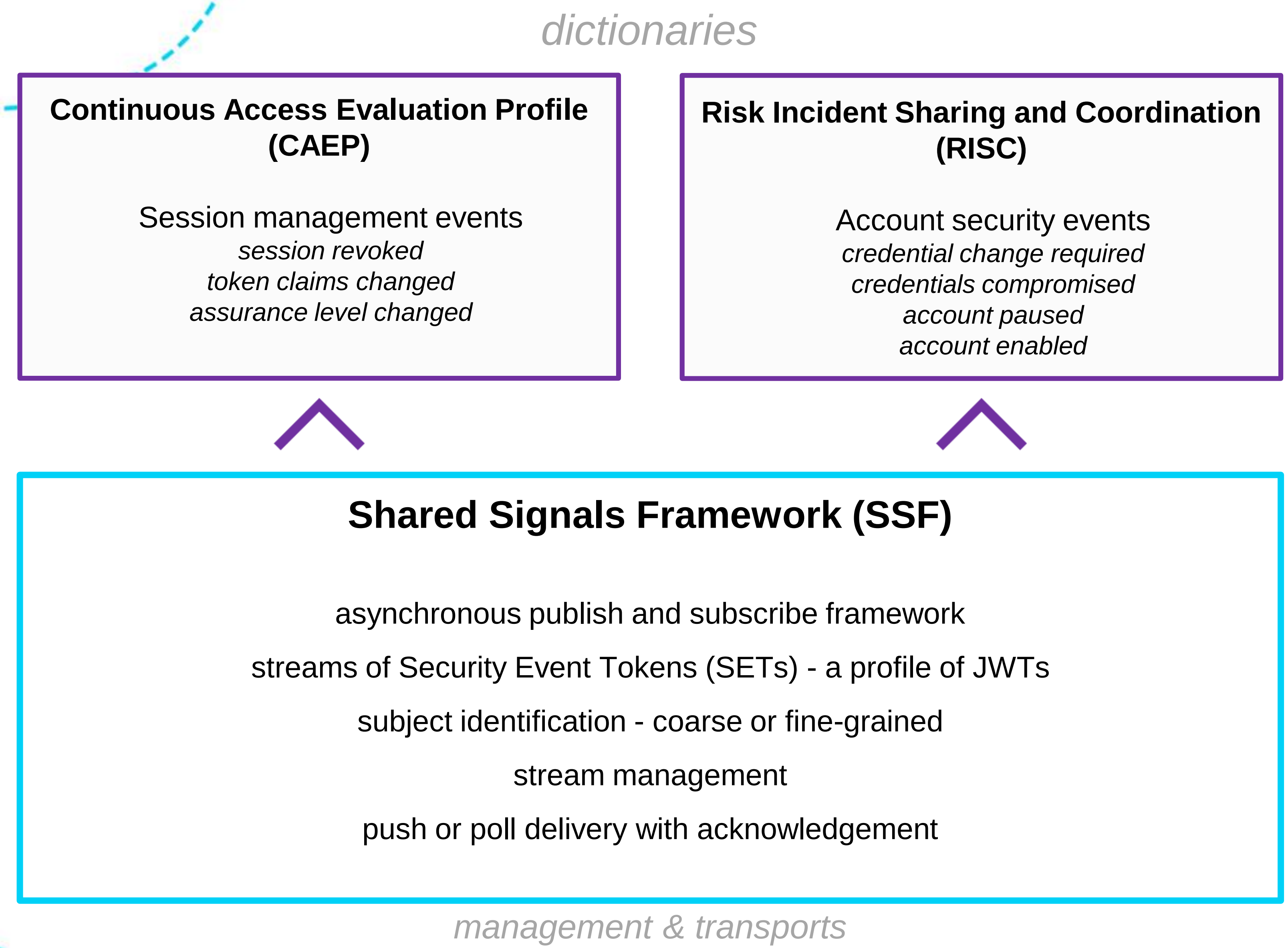
- Problem Statement
- SSF & CAEP Overview
- Deeper Look at SSF
- Demo
- Specific Use Cases

The Zero-Trust Security Problem

- Users simultaneously logged in to hundreds of services
- Independent sources of truth for various information:
  - Device compliance and security
  - User credentials and authentication
  - User authorization
  - Compromised Credentials
  - Behavioral analytics

How SSF and CAEP Help Zero-Trust Security

- Multi-vendor interoperability is of paramount importance
  - Reduced security without interoperability
- SSF conveys information that affects session security to where it is needed
- Efficient and near real-time
- CAEP: Non-prescriptive session-related events



dictionaries

Continuous Access Evaluation Profile (CAEP)

Session management events

session revoked
token claims changed
assurance level changed

Risk Incident Sharing and Coordination (RISC)

Account security events

credential change required
credentials compromised
account paused
account enabled

Shared Signals Framework (SSF)

asynchronous publish and subscribe framework

streams of Security Event Tokens (SETs) - a profile of JWTs

subject identification - coarse or fine-grained

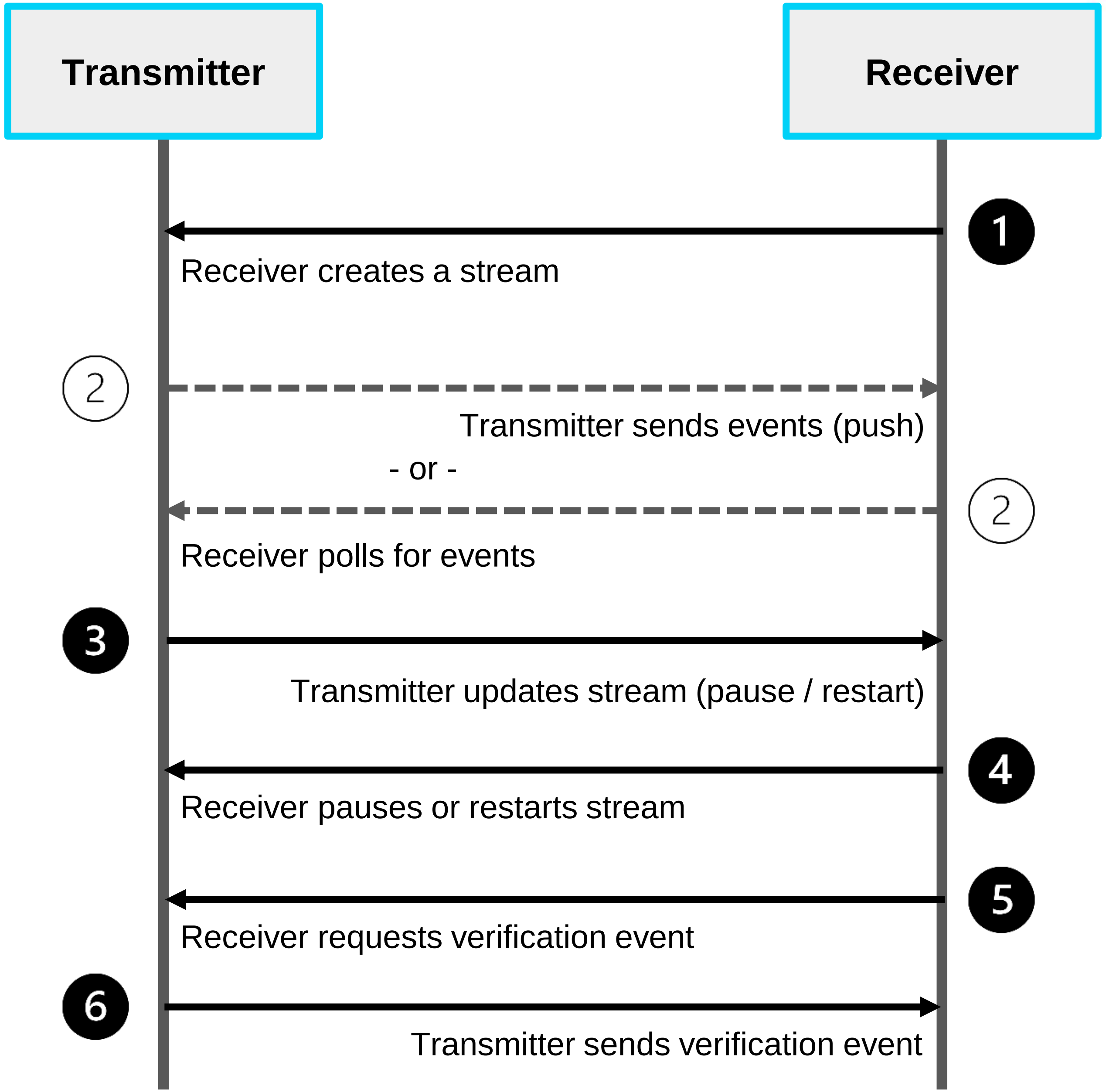
stream management

push or poll delivery with acknowledgement

management & transports

Stream Controls

- Event types are negotiated during stream creation
- Push and poll delivery methods
- Verification events to check liveness of the stream



Transmitter

Receiver

1 Receiver creates a stream

2 Transmitter sends events (push)

- or -

2 Receiver polls for events

3 Transmitter updates stream (pause / restart)

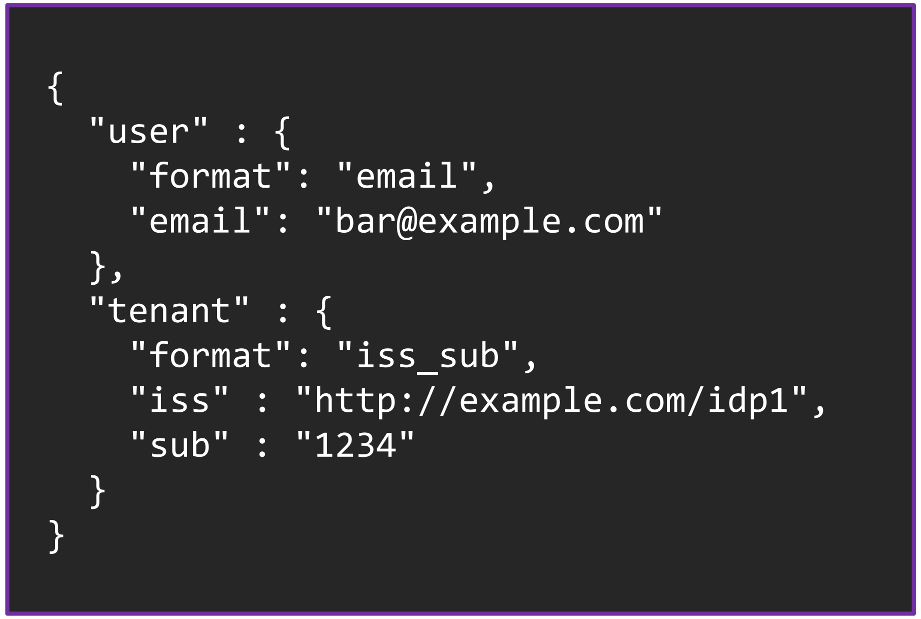
4 Receiver pauses or restarts stream

5 Receiver requests verification event

6 Transmitter sends verification event

Subjects in SSF Events

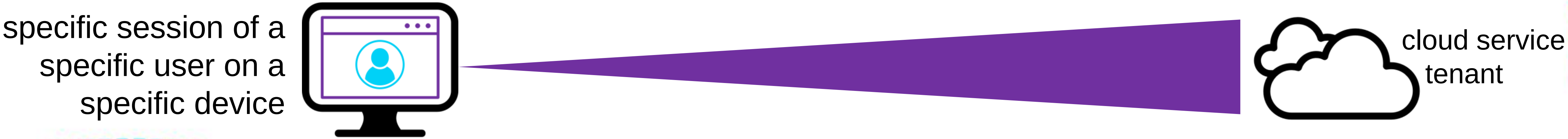
- **Simple** subjects: email, phone number, unique identifier, etc.
- **Complex** subjects:



```
{
  "user" : {
    "format": "email",
    "email": "bar@example.com"
  },
  "tenant" : {
    "format": "iss_sub",
    "iss" : "http://example.com/idp1",
    "sub" : "1234"
  }
}
```

More on Event Subjects

- Specific subjects may be added to or removed from streams
- Authorization may be user-specific
- Subjects may be implicitly included in streams
- A subject value always relates to one principal, but it may be coarse-grained or fine-grained



caep.dev Demo

Session Revocation Use Case

1. User begins federated session with a Service Provider (SP), using a login from an Identity Provider (IdP)
2. SP adds user to SSF stream with IdP, creates one if the stream doesn't exist
3. IdP terminates user's session
4. IdP sends "session-revoked" CAEP event to SP over the stream
5. SP takes corrective action (typically revokes its session)

Device Management Use Case

- A device management service acts as a SSF Transmitter, and an IdP acts as the SSF Receiver
- If a user’s device falls out of compliance, the device management service sends a “device compliance change” CAEP event to the IdP
- The subject of the event is an identifier known to both the IdP and the device management service
- The IdP takes corrective action (e.g. send session revocation events to all SPs)

Authorization Use Case

- SP (SSF receiver) accepts resource identifiers in login tokens. A logged in user only has access to these resources
- IdP (SSF transmitter) incorporates resource identifiers into tokens based on custom business logic
- When a change occurs at the IdP (e.g. user no longer needs access to a specific resource), IdP sends a “token-claims-change event”
- SP takes corrective action (removes user access to that resource)

Call to Action

- Explore the standards at https://openid.net/wg/sharedsignals

- Get familiar and use open-source from: https://sharedsignals.guide

- Test your receiver implementations and learn about the standards at https://caep.dev

THANK YOU!