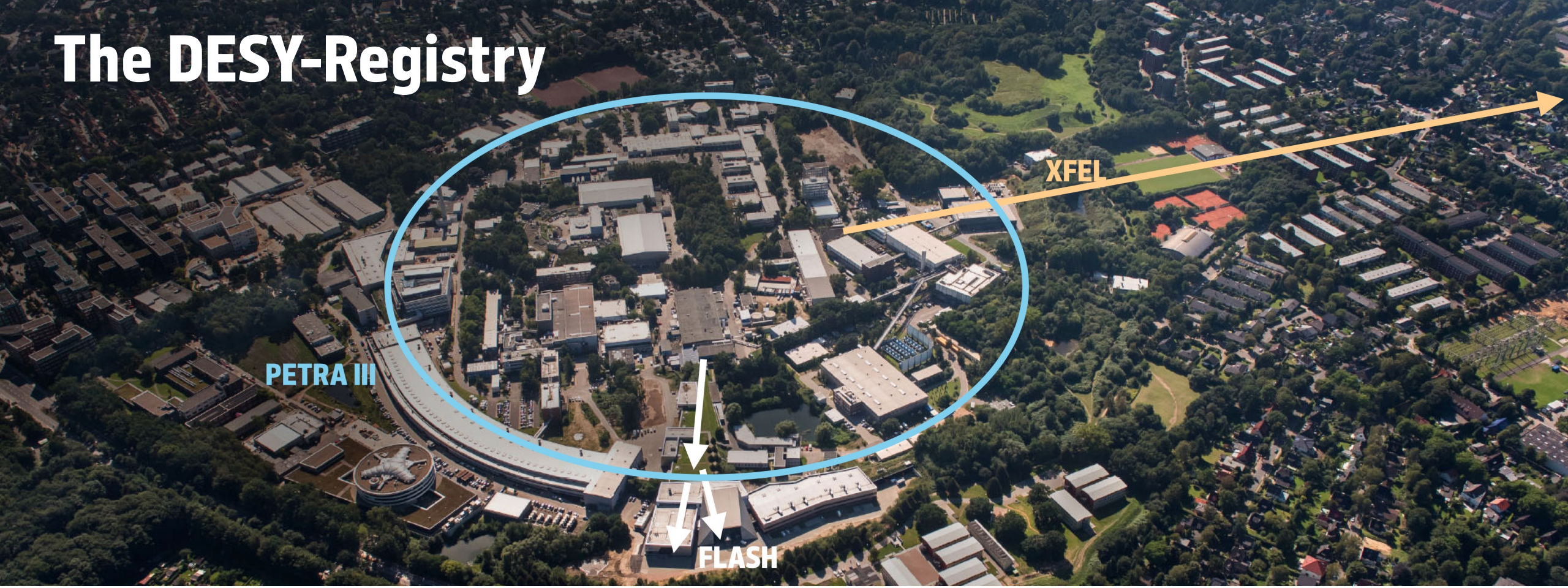
The DESY-Registry



HEPiX autumn 2023
University of Victoria
British Columbia
Canada

Daniel Knittel
Dirk Jahnke-Zumbusch

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

HEPiX Online

DESY.

Brief overview – 1

Provisioning of accounts, groups, resources, …

- provisioning system
  - for accounts across centrally administered systems
  - and groups
    - used for setting s ACL in
      - file systems,
      - shared folders on Zimbra etc.
  - also group-memberships
    - incl. groups in groups
    - integrating now hundreds of beam-time groups to avoid / free up reserved id-blocks
  - and resource access
    - e.g. to UNIX, AFS, workgroup servers, Oracle, mailbox, sync & share (Nextcloud), …
    - may also be automatically assigned
    - parametrized e.g. login shell, displayName, mailbox class

- directly manages 12 systems (platforms)
  - Kerberos-5 (Heimdal)
  - Windows-AD
  - LDAP-service(s) (general, EPICS)
  - Zimbra mailboxes and other e-mail addresses / entities
  - and more (AFS (yes) Sync and Share, Oracle-DBs, Device-Access, …) plus indirectly via AD/LDAP

- administrative entities
  - Namespaces ~ DESY departments
    HR, Procurement, Directorate, Part. Physics, IT, …
  - roles with DESY-Registry (R2)

- pre-pandemic: ~3.000 guest scientists yearly
  European-XFEL guests also appear in the Registry

Brief overview – 2

What is *not* administered and how came everything into place

- not: person database
  - Registry only concerned with accounts, groups, …
  - Registry uses already established in-house solution →IAM (identity and access management)
  - IAM implemented using Oracle database
  - WebUI implemented with Oracle's APEX

- Belle II's membership management system B2MMS is technologically seen predecessor of DESY's user registry
  - uses Oracle as a database
  - and Oracle APEX as WebUI
  - implemented using micro services (that was new in comparison to person database)
  - => *gave confidence when starting R2 implementation*

- the "old registry" R1 was productive from 2005 on
  - also an internally developed software
  - based on JBoss / Tomcat application

- R2 is an evolution of R1 and based on B2MM2 tech.
- production start with parallel operation R1||R2
  - DESY-Registry (R2) pre-production 22.2.2021
  - first: users and admins were still using R1-WebUI
  - generated R1-events, were directed to R2
  - R2 generates its own events, being sent to platforms
  - second: users and admins using R2-WebUI since November 2022
  - => *this helped a lot in comparison to a "all at once on one day" switch-over of event mechanism and WebUI (plus API)*
  - => *took a long time, starting in Nov. 2016*

Brief overview – 3

software being used

- mainly based on Oracle database components
  - APEX for the Web-UI => *eases UI development*
  - own database instances for production and development; database group's responsibility => *support/updates by them*
  - own single database instances for testing new stuff (big changes, new versions of APEX or Oracle-DB) *=> maintenance work very limited*
- micro-services
  - data hosted by separate Oracle schemes (users)
  - communication via a channel (common design pattern, in-house implementation)
  - called within Oracle with Oracle means

```
cmd_obj.add_parameter('action_name','get_acct_attrs');
cmd_obj.add_parameter('account_id',12345);
communication_channel.execute(cmd_obj);
-- get results
p := cmd_obj.get_parameter(…);
```

  - REST-API, concept for general usage, specifically EXFEL Oracle REST Data Services
  - in the end: generate & queue events i
- events fetched by platform adapters
  - events carry a priority value, so PAs may (should) honor priorities
  - results as feedback with return codes and text (0-ok, 400-temp fail, 500-failure)
  - stubs for Python, C#, PERL, PL/SQL *=> platform admins have to implement not everything themselves and may use a familiar prog. language*
- person data is cached
  - on special request of the database group no direct usage
  - data is stored in encrypted tablespaces
  - birthday data is hashed
  - checksums to trigger updates by row
  - regularly scheduled sync and manually triggered single sync
  - *=> better code performance*
  - *=> complex handling (and sometimes debugging)*

Brief overview – 4

patching and adding functionality

- APEX can easily be updated; instantly available
  => *very convenient*
- web application may be exported as a PL/SQL program
  => *this is text-only, git check-in easy*
  => *on-the-fly import into production instance easy*
- R2 core: PL/SQL code within database has its caveats
- PL/SQL code chunks are split into
  - data definition part (tables, views, sequences)
  - code specifications (declaration of public F/P, constants, types)
  - code bodies (code logic of all functions and procedures)
- => *may be managed in Git*
- => *altering PL/SQL has to be done in a specific way, e.g.*
  - *declarations must come first*
  - *new table column: must use ALTER TABLE DROP/CREATE is no alternative*
- updates, patching of R2 code adds some extras
- R2 runs in Oracle RAC
  - multi machine cluster
  - all code pieces running in parallel
- altering packages mark still running packages as "stale" (CREATE OR REPLACE TYPE …)
  - => *this persists as long as an APEX web session still sticks to that package*
  - => *error messages in web UI show up*
- workaround:
  - package gets a new name …_v2
  - microservice re-routing
- Oracle's EBR does (did?) not address this in 19c

Requests for enhancement – 1

New WebUI, new feature requests

- shortly after going productive a workflow mechanism was requested
  - user agreements => implemented
    (VPN, private usage of DESY resources)
  - user != account => located better in person database, but changing password regularly (still) is a good point to get an agreement after 6 moths at the latest
  - moving accounts between namespaces of different admins groups
- bulk services
  - most basic available at start (group memberships)
  - setting password for computer course accounts
  - changing resource attributes, e.g. login shell, printer (meant for platforms)
- exchange mechanism of secrets
  - short text (4000 characters), encrypted, notifications, automatically deleted
- REST-APIs for other guest services, e.g. DOOR
- reports – separate APEX app is planned
  - more complex / history / status quo at point in time
  - it's a database, but …
    - minimizing grants (SELECTs) across database schemes
    - micro service architecture
    - use of R2 roles for access control, no DB-inherent ACLs
- R2 will be able to create person records in persons database
  - before account creation owner must be in person cache
  - currently no creation of person records
- additional workflows
  - users may request resources, admins approve
  - omitted in the first version => *would have postponed start*
  - will have a look at the (relatively new) APEX workflow

Requests for enhancement – 2

more feature requests

- multi factor authentication (MFA)
  - implemented since day 1: TOTP
  - later SMS, external e-mail address
  - in-house implementation, on-premise data, not exposed
    => then rolling out MFA generally became more urgent
  - R2 now uses external TOTP solution (Privacy Idea)
    - creation
    - used for authentication
  - login to password.desy.de enforces usage of MFA
    VPN / ssh / mail requires this anyway
    => use of stolen credentials not that easy any longer
  - all users are encouraged to add at least two second factors
    => be prepared for large scale password resets
- management of federated users
  - what are the connections between those and the local entities (OS users, LDAP users, …)
  - find out requirements
- more consistency checks
  - how should the services look like …
  - … and what's observed in reality

Requests for enhancement – 3

connections to e-mail services

- improving data distribution to e-mail platform
- methods used during development: ssh and scp
- will be replaced by proper API

- message queuing foreseen (RabbitMQ)
  - no direct commands
  - be more generic, e.g. jobs like “create a mailbox w/ following params…”
  - persistence of messages in case of connectivity issues
  - RMQ has its own complexity (HA etc.)

- new mailbox backend?
- two endpoints to be addressed:
  cannot be Zimbra’s “zmprov”-command, must be generic, e.g. “set mailbox class to outlook_enabled”
- moving between backends would be new
- control of migration must be implemented
  - hold traffic for target address xxx@desy.de
  - create mailbox
  - transfer data from A to B
  - unhold traffic
  - clean up

Summary
DESY Registry R2

- in-house implementation: flexible, took time
- decision for Oracle/APEX:
  rely on an extensive framework
  - confidence in feasibility
  - UI easy to implement and easy to patch
  - REST services framework available
  - infrastructure driven by database group
  - upgrades in R2 core a bit tricky due to PL/SQL
  - and extra tricky due to APEX session handling
  - combination of Oracle/APEX is a bit of a niche
    → knowledge not widely spread (but at least at DESY)
- Java-based Quarkus would be an alternative
  "A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards."

- R2 architecture: flexible configuration
  → new functionality w/o the need of programming
- extensible architecture
- delivery of platform adapter stubs eases
  PA development
- adaptations and bug fixes completely in our hand
  → more a pro than a con
- going into production by parallel operation helped a lot
- good acceptance of admins and users
- lots of feature requests
  - prioritised and addressed one after the other (not all)
  - namely: workflow, report application, federation, data distribution, mass PW reset by use of MFA
- constant maintenance efforts required

Thank you for your attention

Q & A