Evolveum

Tasks Webinar

Slavek Licehammer, October 2023
Identity Governance Strategist

Agenda

- Tasks introduction
- Tasks vs. activities
- Types of tasks
- Error handling
- Parallelization

Tasks introduction

- Logical unit of work
- Synchronous / asynchronous
- Single-run / scheduled
- Single-thread / parallel

Evolveum

Task vs. activity

Activity

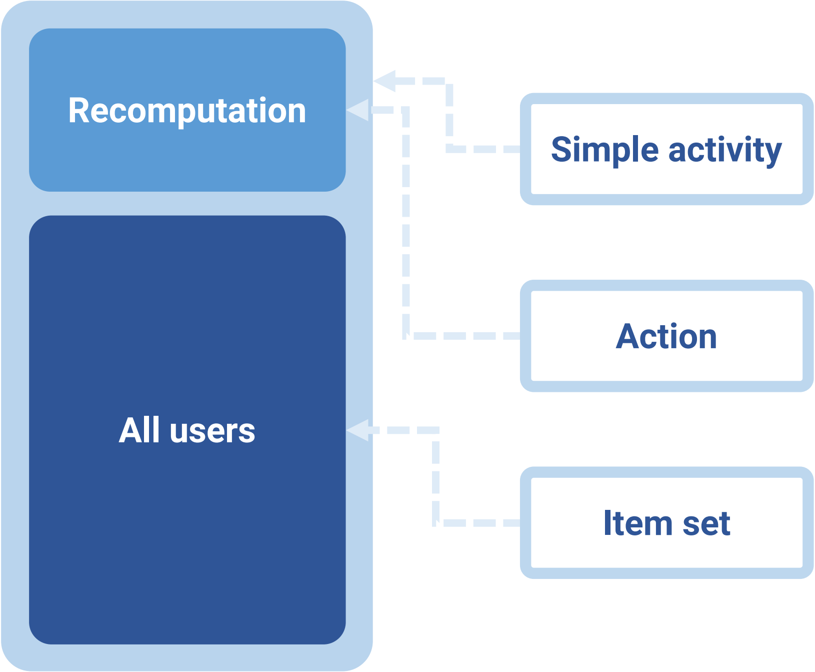
- **Work to be done**
- What action
- What objects
- Distribution
- Error handling
- Reporting

- Can be distributed to multiple worker tasks

Task

- **Execution wrapper**
- Execution status
- Reporting results
- Scheduling
  - Time
  - Nodes

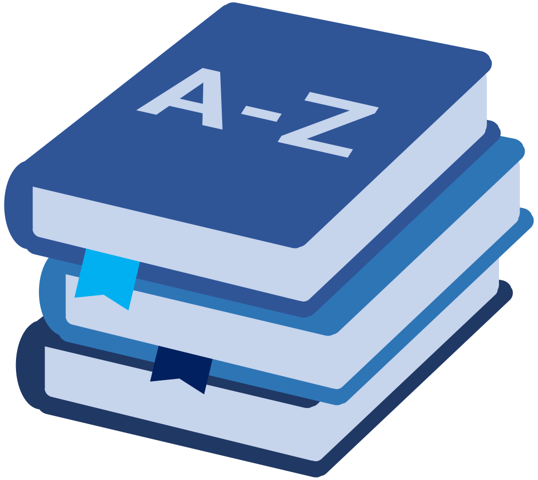
- Can contain multiple activities



Evolveum

Tasks vs. activity terminology

- We are often referring to tasks by the activity
- Reconciliation task
  - Tasks with reconciliation activity



Task execution status

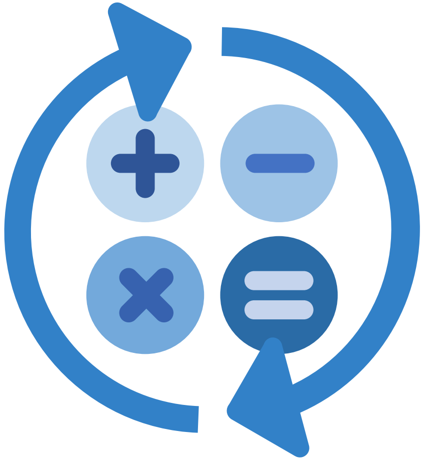
- Runnable
- Running
- Waiting
- Suspended
- Closed

Provisioning and synchronization

- Provisioning is “included” (no separate task)
- Import
  - Iterate over resource objects
  - Synchronize each of them (include provisioning)
  - Changes are immediately provisioned
- Reconciliation
  - Pending operation + import + remaining shadows clean-up
  - Purpose – make resource consistent with midPoint data
- Live-sync
  - Synchronize changed objects (include provisioning)

Recomputation

- Similar to reconciliation
- Iterate over objects in midPoint
- Synchronize then with all resources
  - Inbound & outbound
- Primary purpose - make sure midPoint data are consistent
  - Provisioning is “a side effect”



Evolveum

Configuration

- Everything is configurable
- Watch synchronization configuration
  - For each resource
- Tasks are only executing what is configured
- Dependencies
- Limiting propagation
- Reconcile all

Provisioning propagation task

- Immediate provisioning is not always desirable
  - Grouping changes together
  - Slow resource
  - Schedule load on resource
- Configure resource to postpone provisioning
- Create the provisioning tasks

Evolveum

Other tasks

- Validity scanner
  - Evaluates validFrom and validTo
  - Uses last scan time-stamp
  - Can run in parallel
- Trigger scanner
  - Executes triggers
  - Can run in parallel
- Shadow refresh
  - Pending operations on shadows
  - Dead shadows clean-up

Scripting tasks

- Used to execute custom (bulk) action
- Non-iterative scripting task
  - Arbitrary script
  - Not recommended
- Iterative scripting tasks
  - Script + input definition (query)
  - Recommended approach
  - Parallelization
  - Reporting progress

Error handling

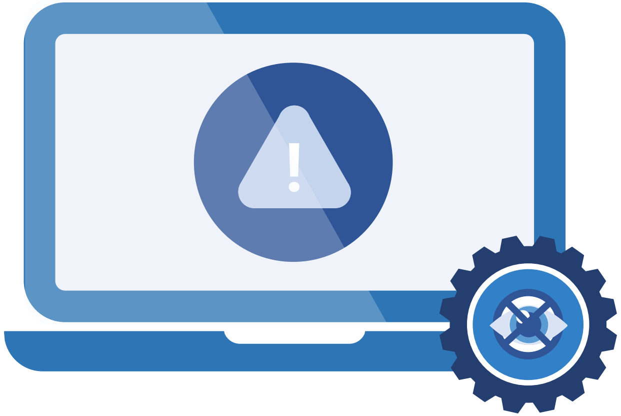
- Errors affecting the whole activity
  - E.g. Resource down
  - Not much that can be done
  - Task is suspended
  - Best strategy is to examine and reschedule
- Errors affecting individual objects
  - Several options for automated error handling

Error handling

- Situations
  - Partial / fatal error
  - Category (network, security, policy ,…)
  - Depending on connector support
- Reactions
  - Ignore (usually default)
  - Stop (default for live-sync and async. update)
  - Retry later (only for synchronization activities)

Error handling – Ignored situation

- Will be solved in next run (e.g. reconciliation)
  - Monitoring is recommended
- Re-run for failed objects
  - Recover task needs to be configured
    - Different task type supported
  - Failed object selector



Evolveum

Error handling – Stop situation

- Requires manual intervention
- Usually live sync. or async.
- Fix problem and resume task
- Fix problem and objects (reconciliation)
  - Reset live sync token
  - Clean async. messages queue

Error handling – Retry later situation

- Limited to shadow object and synchronization tasks
- Simple configuration – only scheduling
- Carried out by a trigger scanner
- Recommended if applicable

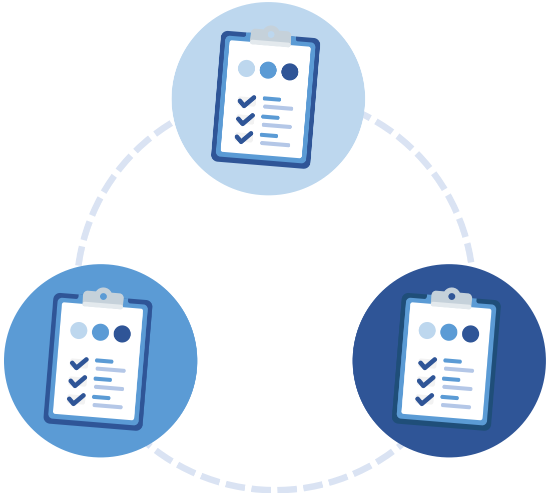
Evolveum

Parallelization

- Sometimes you need to scale…
- Multi-threading
  - Live sync
- Multi-node tasks
  - All supports also multi-threading
  - Recompute, reconciliation, import, iterative scripting, validity scan, trigger scan, shadow refresh, propagation

Multi-threading

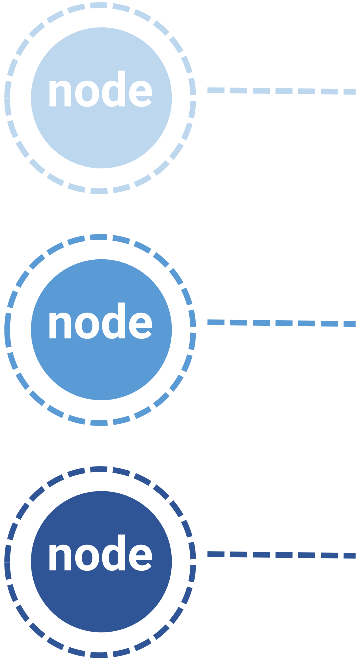
- Works on a single node
- On multi-node thread are on each node
- Simple configuration → number of threads (per node)
- Configured in activity distribution section
- Not supported for all activities
    - Check documentation



Evolveum

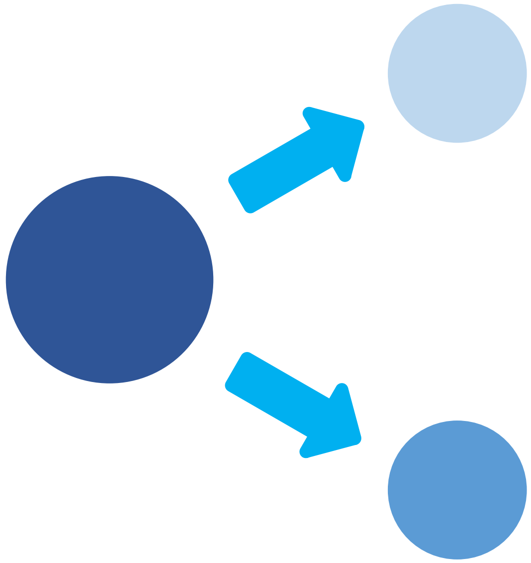
Multi-node parellelization

- Works in combination with multi-threading
- Distributed to nodes using buckets
- Auto-scaling support
  - Simple parameter (enabled/disabled)
  - Changing nodes during task run is not supported
- Not supported for all activities
  - Check documentation



Bucket definitions

- Split objects into roughly equal size sets
- Segmentation
  - Numeric, string, OID, explicit
- Each node is querying objects defined by the bucket
  - Connector support
- It’s good to know your data



Evolveum

Parallelization recommendation

- There is no silver bullet
- You need to know you environment
- Bottle necks
  - Thoughtput
  - Slow resources
- Set your scaling accordingly
- Auto-scaling in combination with external monitoring

Conclusion

- This was tasks overview
- Details are in documentation
- Many options were not covered
    - Tasks chaining, various synchronization options, …
- Feedback is welcome

Evolveum

Thank you for your time

Do you have any **questions**? Feel free to contact us at **info@evolveum.com**

Follow us on social media or **join us** at GitHub or Gitter!

in /Evolveum

@Evolveum

/Evolveum

/Evolveum

/Evolveum

/Evolveum

Evolveum