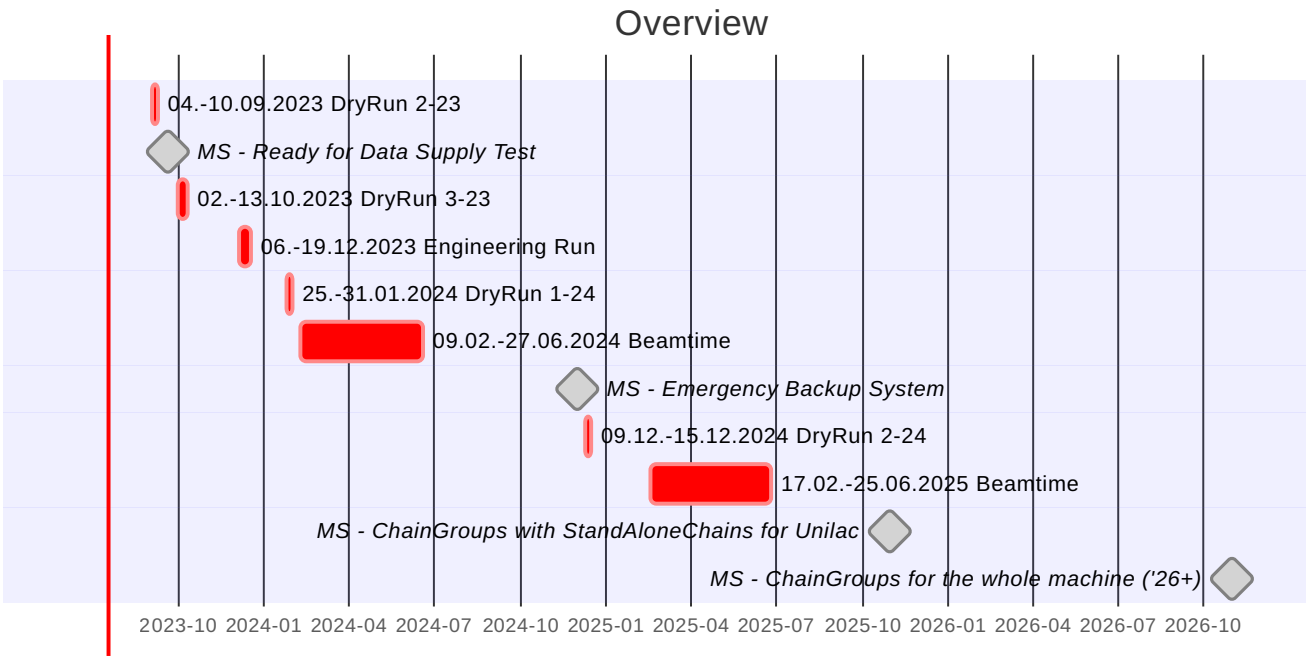
Unilac Workshop 19.07.-20.07.2023

Development Steps

- Development 2023 - Data Supply Test
- Development 2024 (Beamtime '25) - Emergency Backup System
- Development 2025 (Beamtime '26) - ChainGroups with StandAloneChains for Unilac other Machines still use PatternGroup and Patterns
- Development 2026+ - ChainGroups with StandAloneChains for the whole Machine



Development 2023 - Data Supply Test

Assumption

- Unilac PZ still exists and is used for timing etc.
- LSA only resupplies/overwrites the device data for a specific Unilac accelerator to verify that key devices can be supplied with correct data from LSA

Requirements

- Needed Unilac devices are adapted for LSA Data Supply (removal of complex types)

- Needed Unilac devices are imported into LSA (need to be in FESA DB, calibration curves in CDB, etc.)
- Unilac Model for devices under test is finished
- If devices using WR timing are tested, we need timing bridges that translate the events from MIL -> WR Timing appropriately

LSA

Two options:

1. Use a "Ghost"Pattern without Group, hardcoded to a Test Unilac Accelerator
2. Unilac PatternGroup with one Pattern, hardcoded to a Test Unilac Accelerator

Development 2024 (Beamtime '25) - Emergency Backup System

Assumption

- One beam at a time, no full parallelization possible
- No major change in the current implementations, only minor adaptations for Unilac
- SIS18 -> Unilac request still “ad-hoc” on request like today (through UNI-PZ Gateway?)
- All Unilac devices are adapted for LSA Data Supply
- All Unilac devices are imported into LSA
- Unilac Model for devices is finished
- Ion Source Timing can be supplied (no concept in LSA)

Requirements

- Unilac Timing <-> WR Timing Bridge is still used
- Dedicated Timing Master that supports 50 Hz sync
  - Single Thread probably enough
  - 15 edges for the default pattern, since we probably want 15 patterns and minimal implementation changes
- UNI-PZ Gateway or something else takes care of “Signaling” a request from the SIS18 DM to the Unilac DM
- Timing bridges translate the events from WR -> MIL Timing appropriately
- BSS supports a Unilac PatternGroup / Pattern Graph that can be supplied to the Unilac DM
  - Generated Unilac Pattern Group should support 15 patterns
  - Stopping a Unilac Pattern Group should (probably) finish the Pattern Group

    - Changing the Unilac Pattern Group should not stop the Unilac e.g. write new pattern, switch edge, delete old pattern (or similar functionality that achieves the same)

LSA

- Creation of a Unilac PatternGroup that is explicitly written into the Unilac DM
- The Unilac PatternGroup contains max 15 Patterns incl. placeholder patterns that are played round-robin
    - Pattern - Beam, the “real” timing and settings that should be used
    - Pattern - No Beam, a Dummy Pattern to waste time to fit the Ion Source repetition rates (x 20ms set by repetition rate), reduction (untersetzung), etc.
- Clarify and Implement scheduling related questions
    - Dummy Patterns
        - What does a dummy pattern do?
        - Would one dummy pattern be enough?
    - How to handle repetitions?
    - Do we need to generate an appropriate default schedule (warmhaltepulse) ?
- Only use / force ByPass Trim for Unilac
    - on timing change use BSS to update the relevant pattern and wait until the old pattern is deleted (bss blocks until deletion or until a timeout is reached)

Development 2025 (Beamtime '26) - ChainGroups with StandAloneChains for Unilac other Machines still use PatternGroup and Patterns

Assumption

- Everything is controlled using DMs, no PZ anymore
- For Unilac LSA only generates Chain Graphs, Scheduling these Graphs is moved to BSS
- Move to Oracle 21 - not clear if Acc6 Instant Client Driver is still working

Requirements

- One Timing Graph for the whole facility that is supplied to the DMs, or something similar to “edges” between UNI-DM and RING-DM
    - so we can configure edges to start e.g. SIS18 graphs from the Unilac
    - so we can have “control graphs” that synchronize the runtime behaviour and “fork” other graphs
- Additional Timing Graph Functionality

    - Creation of threads (basics should be already there from the Unilac Booster implementation)
        - Using the next free / available thread atomic
    - Start / Fork execution of threads
    - Joining of threads, depending on the (re)implementation of some SRM features
    - Thread local storage of some values
    - Cleanup of “old” unused graph pieces (to be discussed how to handle)
- BSS needs an additional endpoint supporting ChainGroups and Chains
- Application using the famous Peter-Gerhard-Algorithm to generate a Chain Group Scheduling

LSA

- Support for StandAloneChains
- Discuss and implement new interface to BSS Scheduling
    - Context Graph & Schedule Graph
    - Signals, also use by-name convention for CGs/Unilac or more explicit?

Development 2026+ - ChainGroups with StandAloneChains for the whole Machine

Assumption

- Patterns and PatternGroups are removed, everything is moved to ChainGroups / StandAloneChains
- LSA only generates Chain Graphs for everything, Scheduling these Graphs is moved to BSS / Scheduling-App

Requirements

- Apps are ChainGroup-aware
- Timing Master “cluster” available
- BSS needs more structural Chain information
    - to decide which chain groups can run in parallel (e.g. Unilac + Cryring Injector)
    - to switch chaingroups on request (e.g. storage ring mode / runtime control)

LSA

- Remove Patterns & PatternGroups
- Generate injection / extraction information / markers
- Add StandAloneChains & ChainGroups features
  - support for Synchrotron Mode
  - support for Storage Ring Mode
- Eventually adapt LSA<->BSS interface
- Switching Chain Groups
  - straight
    1 switch on -> Ramp up devices once to inter cycle level.
    1 operate machine
    1 switch off -> Ramp down devices
  - slanted - Pre / Post / “Basic Unilac Filler Chain Group” for switching Chain Groups on Request

LSA - other

- Development 2023
  - continue RMI -> REST interface migration
  - RBAC test
  - New Particle Interface Prototype
  - Move LSA Test DBs to OracleXE containers (effort regarding Containerization / Kubernetes, Testing)
- Development 2024
  - complete RMI -> REST interface migration
  - work on LSA updates / notification using RDA3
- Development 2025
  - Move to Oracle 21
- Development 2026+
  - Focus on SIS100 concepts
  - Scheduling 2.0 (chain from SIS18 to SIS100)
  - SubChains that not span all particle transfers (e.g. booster SubChains only span SIS18 and not span all particle transfers up to SIS100)
  - Work on “beam” (which chains inject into which chain)